Compact Series CPLD configuration user guide

(UG030004, Version 1.5)
(2020.4.24)

Document version revision record

VERSION NUMBER	RELEASE DATE	REVISION HISTORY
V1.0	2018/8/29	initial version
V1.1	August 20, 2019	1. Update Figure 3 Download Configuration Timing 2. Delete the embedded Flash capacity information, see "UG030006_Compact Series CPLD Embedded Flash (EFlash) User Guide 3. Note that the TCK of the JTAG operation flow is 50MHz 4. Update the contents of the appendix, some operations need to wake up the embedded Flash in advance 5. Improve the description of indirect programming of embedded Flash in the system, add logic block diagram 6. Improve the timing diagram in the appendix and indicate the detailed timing 7. Added the embedded Flash operation process for 4K/7K devices 8. Improve the description of the dual-boot function, and add the dual-boot function of the main self-loading supported by the D-type device 9. Improve the description of the feature control bits according to different devices 10. Add JTAG instruction description 11. Add description of configuration register
V1.2	2019/11/15	1. Updated appendix 2, corrected the process of "programming bit stream" and "programming user Flash" 2. Update Appendix 3 3. Updated Appendix 4 to add a complete timing diagram for reading UID
V1.3	1/6/2020	1. Modify the embedded Flash page address to correct the 4K embedded Flash address error. 2. Modify Table 29 to delete the description of "x4" in the watchdog register. 3. Increase the dual-boot operation process. 4. Increase the introduction of the embedded Flash storage structure, and increase the abnormal operation prompt. 5. Add description, the I^2C interface should be canceled in user mode Enable OSC Shut Off. 6. Add JTAG, slave SPI, slave I^2C reference application documentation description. 7. Modify Table 13 and Table 14 to add default values. 8. Removed "Adjustment Control Page" description. 9. Change "Hot Start" to "Reset".
V1.4	2020/3/29	1. Adjust chapter layout 2. Add Chapter 7 configuration details 3. Add the complete process of configuring and programming embedded Flash from SPI/I^2C 4. Merge the original slave SPI, slave I^2C process appendix 5. Reflect the busy time after various embedded Flash operations from SPI and I2C in the flowchart
V1.5	2020/4/24	Delete the description of the write UID instruction related to the JTAG interface, slave I^2C interface, slave SPI interface and APB interface

Glossary

CPLD : Complex Programmable logic Device

JTAG: Joint Test Action Group

TAP: test access port

I^2C: Inter-Integrated Circuit

APB: Advanced Peripheral bus

CCS: Configuration control System

SPI: Serial Peripheral Interface

UID: Unique ID

PDS: Pango Design Suite

Table of contents

Figure catalog

Table directory

I. Overview

The Compact series CPLD is a programmable logic device based on SRAM. The chip contains an embedded Flash, which can realize the independent loading of the bit stream from the inside of the chip. Since the configuration information in the SRAM will be lost after the CPLD device is powered off, the CPLD needs to be reconfigured every time it is powered on.

Configuration is the process of writing the bit stream designed by the user into the CPLD, and the entire configuration process is completed by the configuration control system (CCS). Its structure diagram is shown in Figure 1, in which the user logic is connected to 7 slave equipment. The bit stream can be actively obtained by the chip from the external SPI Flash or embedded Flash, or the bit stream can be downloaded to the chip through a host device such as an external processor/controller.

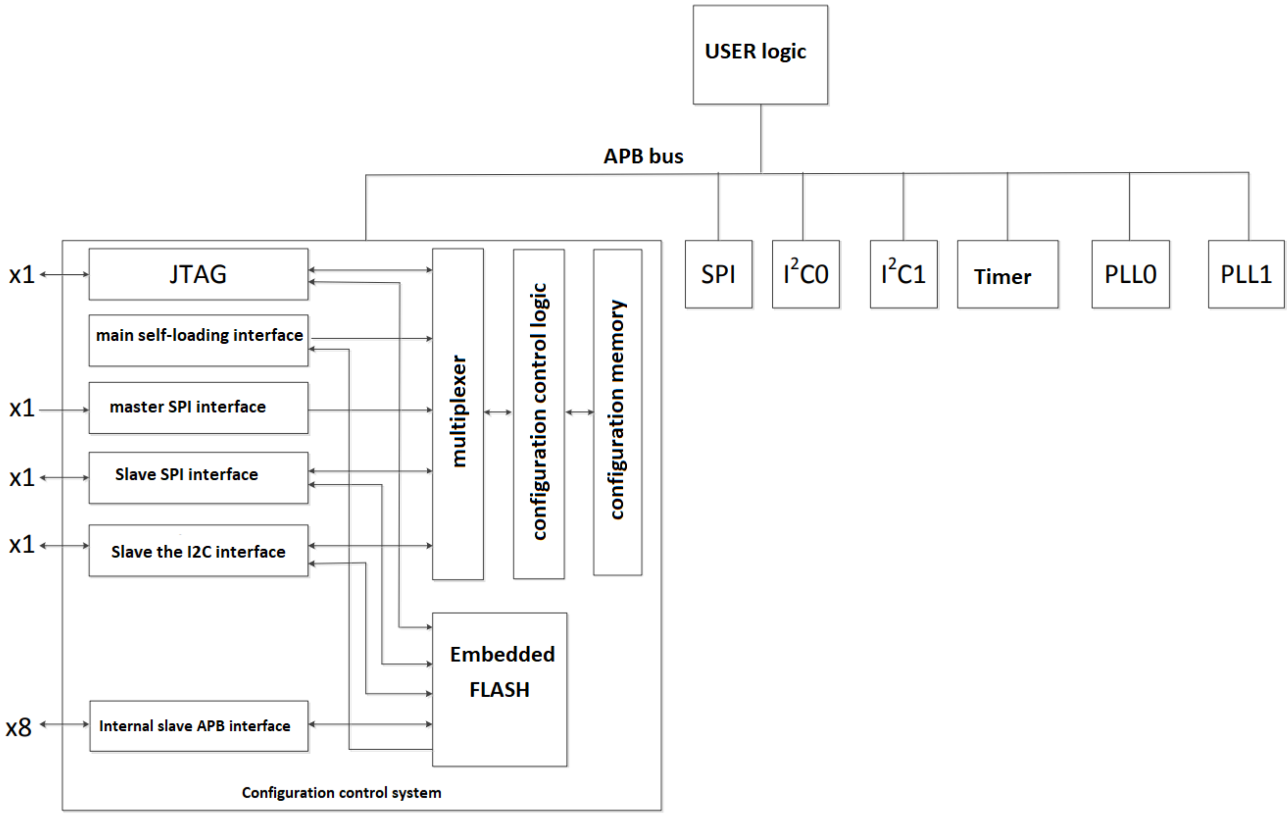


Figure 1 Configuration System Structure Diagram

CPLD devices have 5 configuration modes, the details are as follows:

- JTAG mode, data bit width X1, conforms to IEEE 1149.1 and IEEE 1532 standards, supports configuration and reconfiguration of normal/compressed bit stream, supports readback of normal/compressed bit stream
- Main self-loading mode, support configuration of normal bit stream and compressed bit stream
- Master SPI mode, data bit width X1, support configuration of normal bit stream and compressed bit stream
- Slave SPI mode, data bit width X1, supports configuration and reconfiguration of normal bit stream and compressed bit stream
- Slave I^{2}C mode, data bit width X1, support configuration and reconfiguration of normal bit stream, compressed bit stream

In addition, CPLD devices provide the following features:

- JTAG mode provides a dedicated interface to support online debugging and boundary scan testing
- Watchdog, support configuration time overtime detection
- Support dual-boot, support application bitstream version fallback for all 5 configuration modes
- User logic interface, JTAG user command interface (SCANCHAIN), UID interface, power controller, user wake-up interface (START), internal slave APB interface, readback CRC
- Embedded hard core, including 1 SPI, 2 I^2C, 1 timer

2. Configuration process

The download and configuration process of the CPLD device is shown in the figure below:

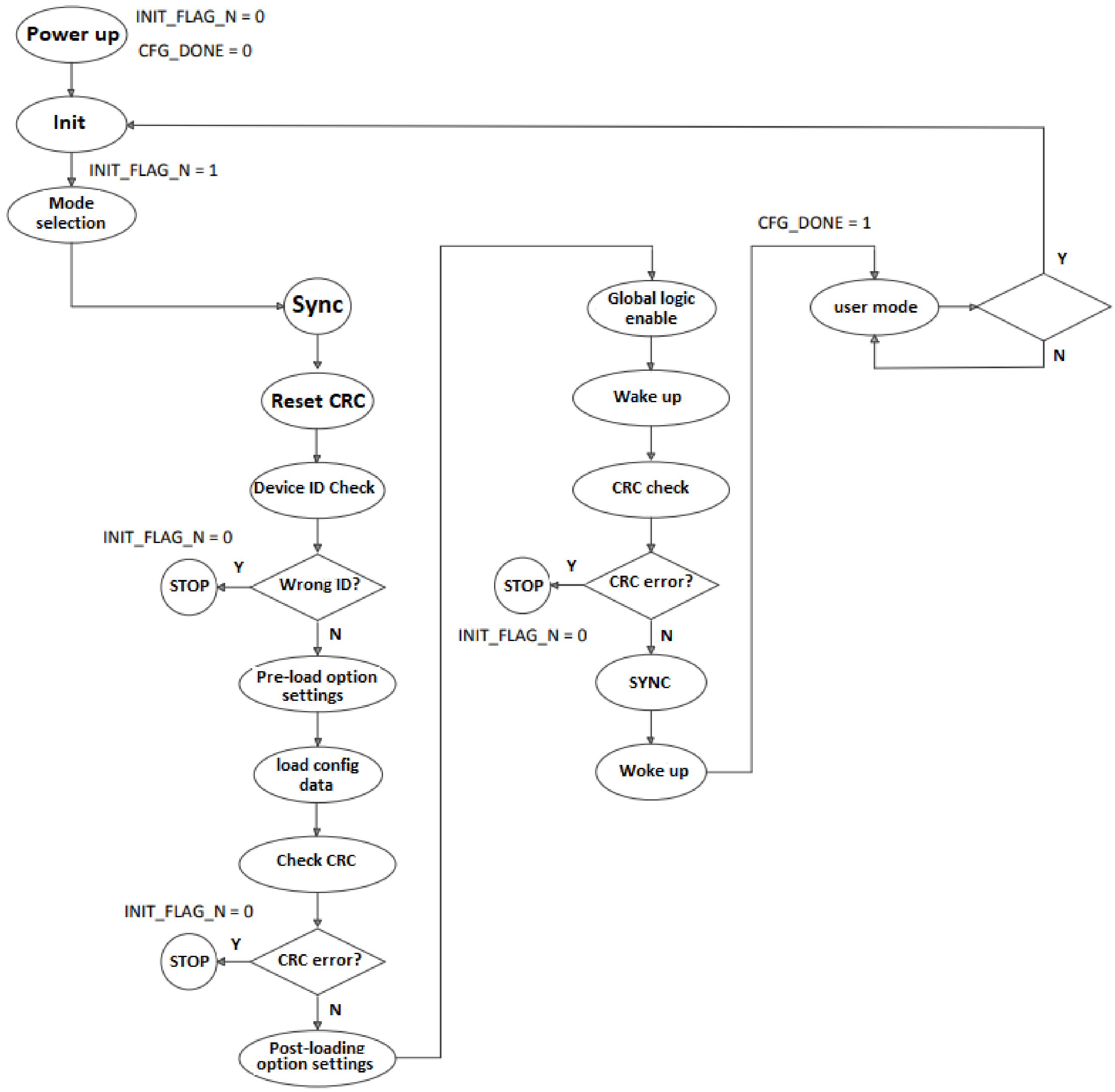


Figure 2 Download Configuration Flowchart

For all configuration interfaces, the basic configuration steps are the same, including setup, bitstream loading, and wakeup. The specific process of CPLD device configuration is shown in the table below.

Table 1 CPLD configuration process table

SET UP			BITSTREAM LOADING							WAKE UP				
Device power up	Device initialization	Configuration mode selection	Synchronize	Reset CRC	Device Identification Check	Option settings before loading	load configuration data	CRC check after loading	After loading option settings	global logic enable	wake up start	CRC check before wakeup	desynchronization	wake

CPLD The power-on configuration timing of the device is shown in the figure below.

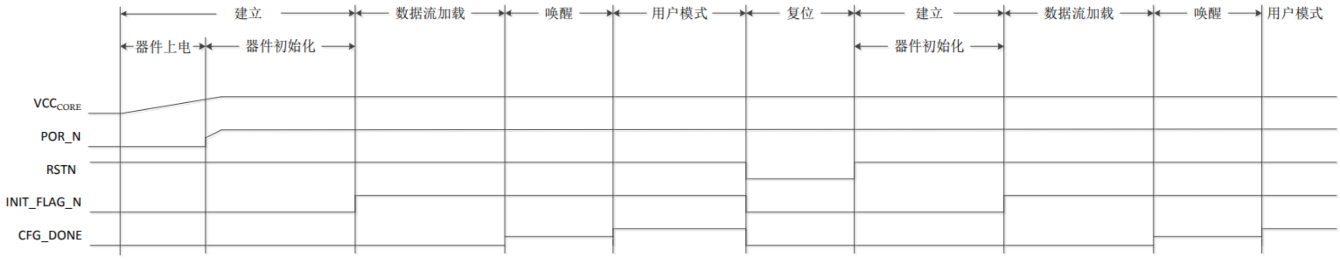


Figure 3 Power-on configuration sequence

(1) Set up

Establish and complete the operation of CPLD device initialization and configuration mode determination, including device power-on, device initialization and configuration mode selection.

1. Device power up

The power-on process of the device is marked by the internal POR_N signal. When the POR_N signal changes from low to high, it means that the core power supply VCC_{CORE} has reached the startup threshold V_{PUP} of the power-on reset circuit.

After the device is powered on, the embedded Flash starts to work normally, and then the configuration control system is released after power-on reset, and the configuration control system starts to work.

2. Device initialization

After the configuration control system starts to work, the device initialization operation is performed first. The configuration control system starts to clear the configuration memory frame by frame. After clearing the configuration memory, the initialization completion flag signal INIT_FLAG_N becomes high. While clearing the configuration memory, the configuration control system powers on the embedded Flash, calibrates and adjusts, reads the feature control bits, reads the UID, and reads the embedded Flash lock flag.

The configuration control system can be kept in the device initialization process by keeping the external INIT_FLAG_N pin low.

When one of functions in Hard Reset, JTAG Instruction Reset, SPI Instruction Reset, I^2C Instruction Reset, Version Rollback is triggered, the configuration control system enters the device initialization process.

3. Configuration mode selection

When the INIT_FLAG_N signal is high, CCS selects the configuration function of the multiplexing pin according to the feature control bit.

(2) Bit stream loading

Different configuration modes have different configuration interfaces. After the configuration mode is selected, the corresponding pins of the corresponding mode are set as configuration pins.

1. Synchronize

The sync word 0x01332D94 is used for alignment on 32-bit word boundaries. After synchronization, the subsequent deframing operation can be performed.

After receiving the correct synchronization word, the configuration control system considers the data to be valid data, and the packet processor of the configuration control system performs the unpacking operation. Any data before the sync word will be ignored.

2. Reset CRC

Reset the CRC register.

CRC checks are performed using a 32-bit CRC algorithm prior to the wake-up operation. When the configuration data is written into the register, the address of the 5-bit register is in the high position, and the 32-bit data is in the low position, which is connected into 37-bit data and a CRC calculation is performed.

CRC check is performed after writing the CRC register data. If the data written into the CRC register is inconsistent with the calculated CRC result, the INIT_FLAG_N signal is pulled low, and the CRC error bit of the STATUSR register is high.

3. Device Identification Check

Check whether the device identification in the bit stream matches the hardware, if not, the INIT_FLAG_N signal goes low, the ID error flag is stored in the status register, and the configuration process is exited.

4. Option settings before loading

Before loading the configuration data, the working state of the device can be specified through the PDS interface:

Watchdog settings

Set whether to reserve the multiplexed configuration port and continue to use it as a configuration port after the configuration is complete

Wake up clock selection

Whether to wait for PLL on wakeup

Master mode clock frequency setting

For the operation method, please refer to "Pango_Design_Suite_User_Guide".

5. Load configuration data

Write a frame of configuration data to configuration memory.

6. CRC after loading an examination

After the configuration data is loaded, perform a CRC check. The CRC value generated by software is stored in the CRC register and compared with the CRC value calculated by CCS. If the two values match, the CRC check passed and all configuration information was correctly written into the configuration memory.

If the two values do not match, indicating that the CRC check failed, the INIT_FLAG_N signal goes low and the CRC error flag is stored in the status register.

7. After loading option settings

After the configuration data is loaded, specify the working state of the device in the configuration register:

Set whether to allow user logic to shut down the OSC

External port security level setting

(3) Wake up

After the bit stream has been loaded and passed the CRC, the CPLD device enters the wake-up phase. The configuration system first enables the logic output of all functional modules inside the CPLD device. Check the CRC again before waking up, and then perform a desynchronization operation, indicating that the configuration operation is over, and subsequent operations need to be resynchronized. Finally, the wake-up is completed, and the corresponding global signals are gradually released.

1. Global logic enable

After the CRC check is passed after loading, enable the logic output of all functional modules inside the fabric.

2. Wake up start

After the global logic is enabled, the wake-up operation is started.

3. CRC before wakeup an examination

Before the wake-up operation is executed, a CRC check is performed. The CRC value generated by software is stored in the CRC register and compared with the CRC value calculated by CCS. If the two values are consistent, it means that the CRC check is passed and the preparation before wake-up is completed. If the two values are inconsistent, indicating that the CRC check is different, then the INIT_FLAG_N signal becomes low, and the CRC error flag is stored in the status register.

4. Sync

Indicates the end of the configuration process. Subsequent operations require a resync.

5. Wake up

After the desynchronization is completed, the wake-up circuit starts to work according to the set timing, gradually releases the corresponding global signal, and finally enters the user mode.

3. Embedded Flash

Pango Compact series CPLD has an on-chip non-volatile storage unit - embedded Flash. In terms of configuration, it brings great advantages for improving configuration efficiency and saving peripheral circuits. Embedded Flash is an important part of Compact series CPLDs. This chapter mainly introduces the storage structure information of embedded Flash, as well as the extremely important storage content—the specific description of feature control bits.

(1) Storage structure

Embedded Flash consists of normal memory and supervisory memory. It is used to store feature control bits, UID, bit stream and common user data. The storage location is shown in the table below.

Table 2 Embedded Flash storage structure

STORAGE LOCATION	STORAGE CONTENT
supervisory memory	feature control bits
	UID
common memory	bit stream or normal user data

Embedded Flash indicates the status through bits [21:18] in the status register. When performing embedded Flash read and write operations, in addition to reading UID, it is necessary to ensure that the embedded Flash is awake and idle, that is, bit [21] busy is 0, bit [19] wake is 1, bit [18] sleep is 0.

Bit [21] busy indicates whether the operation is completed or not. Incomplete operations will cause busy to be high all the time. At this time, the busy bit can only be reset by powering on again. Common incomplete operations mainly include incomplete reading and writing of a page when reading and writing embedded Flash, especially when writing the last page of the bit stream, less than one page should be filled with all 0s or all 1s until the page is complete.

The ordinary memory of the embedded Flash also supports the lock function, and the user can specify the end page of the lock through software or other interface instructions.

After being locked, the content between page 0 and the specified page (inclusive) cannot be read and written, and can only be unlocked by completely erasing the normal memory space.

The locked state does not affect CCS to obtain the bit stream from the embedded Flash.

The storage space of the ordinary memory of the embedded Flash of different devices of the Compact series CPLD is different, as shown in the following table:

Table 3 Embedded Flash general memory storage space

DEVICE	HEAP NUMBER	PAGES PER HEAP	PAGE SIZE (BYTE)	TOTAL CAPACITY (BYTES)
PGC1K	1	332	256	84992
PGC2K	1	332	256	84992
PGC4K	4	320	256	327680
PGC7K	4	452	256	462848

(2) Feature control bit

The feature control bits stored in the embedded Flash determine the specific configuration mode used by the CPLD, the function settings of different configuration modes, and the reserved settings of the configuration multiplexed pins.

Users can select the configuration mode of the CPLD device by setting different feature control bits.

CPLD device supports JTAG, SPI, I2C and internal APB interface to access the feature control bit register in the embedded Flash, so as to change the feature control bit. Users can also modify it through the Fabric Configuration plug-in in the PDS software. For the specific operation process, please refer to the "Fabric Configuration User Guide".

The CPLD device will read the feature control bits from the embedded Flash during the initialization process, and then select the configuration mode according to the feature control bits to enable or disable certain configuration interfaces. Therefore, after updating the feature control bits through JTAG, slave SPI, slave I2C and internal APB interface, it is necessary to reset the command, reset the external RSTN pin (the pin needs to be enabled or reserved) or power on again to make the CPLD device read again. Take and take effect the updated feature control bits. The above configuration interfaces all support the reset command, see the configuration mode description for specific commands.

For 1K and 2K devices, 32 feature control bits form a feature control group CTL, and each feature control bit is descriptioned as follows.

Table 4 1K and 2K Feature Control Group CTL

NAME	BIT	DEFAULT VALUE (BIN)	DESCRIPTION
cfg_msd_en	[0]	0	Master self-loading mode enable, high effective
cfg_rstn_en_n	[1]	1	RSTN in configuration mode Pin enable, active low
cfg_init_en	[2]	0	INIT_FLAG_N in configuration mode Pin enable, active high
cfg_done_en	[3]	0	CFG_DONE in configuration mode Pin enable, active high
cfg_mspi_en	[4]	0	Master SPI Mode enable, active high
cfg_jtag_en_n	[5]	0	JTAG in configuration mode Interface enable, active low
cfg_sspi_en_n	[6]	0	Configuration mode from SPI Interface enable, active low
cfg_si2c_en_n	[7]	0	Configuration mode from the I ² C Interface enable, active low
cfg_spal_en_n	[8]	1	Enable from parallel interface in configuration mode, active low
fallback_en	[9]	0	Dual boot enable, active high
fast mode	[10]	0	Master SPI Fast mode enable, active high
persist_rstn_n	[11]	1	RSTN in user mode Pin enable, active low
persist_init	[12]	0	INIT_FLAG_N in user mode Pin enable, active high
persist_done	[13]	0	CFG_DONE in user mode Pin enable, active high
persist_jtag_n	[14]	0	JTAG in user mode Interface enable, active low
persist_sspi_n	[15]	0	User mode from SPI Interface enable, active low

persist_si2c_n	[16]	0	User Mode from I ^{2}C Interface enable, active low
persist_spal_n	[17]	1	Enable from parallel interface in user mode, active low
i2c_addr[9:0]	[27:18]	1111000000	Chip I ^{2}C slave device 10 bit address
persist_mspi	[28]	0	SPI in user mode Interface enable, high effective
loose_en	[29]	0	slow read enable 0: self-loading, read embedded Flash instruction, read feature control bit instruction, read trim command, the embedded Flash interface adopts normal read mode 1: All embedded Flash Read operations are performed in slow read mode
reserve	[31:30]	-	

For 4K/7K devices, 32-bit feature control bits form a feature control group CTL, and each feature control bit is descriptiond as follows.

Table 5 4K and 7K Feature Control Group CTL

NAME	BIT	DEFAULT VALUE (BIN)	DESCRIPTION
cfg_msd_en	[0]	0	Master self-loading mode enable, high effective
cfg_rstn_en_n	[1]	1	RSTN in configuration mode Pin enable, active low
cfg_init_en	[2]	0	INIT_FLAG_N in configuration mode Pin enable, active high
cfg_done_en	[3]	0	CFG_DONE in configuration mode Pin enable, active high
cfg_mspi_en	[4]	0	Master SPI Mode enable, active high
cfg_jtag_en_n	[5]	0	JTAG in configuration mode Interface enable, active low
cfg_sspi_en_n	[6]	0	Configuration mode from SPI Interface enable, active low
cfg_si2c_en_n	[7]	0	configuration mode from the I^{2}C Interface enable, active low
cfg_spal_en_n	[8]	1	Enable from parallel interface in configuration mode, active low
fallback_en	[9]	0	Dual boot enable, active high
fast mode	[10]	0	Master SPI Fast mode enable, active high
persist_rstn_n	[11]	1	RSTN in user mode Pin enable, active low
persist_init	[12]	0	INIT_FLAG_N in user mode Pin enable, active high
persist_done	[13]	0	CFG_DONE in user mode Pin enable, active high
persist_jtag_n	[14]	0	JTAG in user mode Interface enable, active low
persist_sspi_n	[15]	0	User mode from SPI Interface enable, active low
persist_si2c_n	[16]	0	User Mode from I^{2}C Interface enable, active low

persist_spal_n	[17]	1	Enable from parallel interface in user mode, active low
i2c_addr[9:0]	[27:18]	1111000000	Chip I²C slave device 10 bit address
persist_mspi	[28]	0	SPI in user mode Interface enable, high effective
loose_en	[29]	0	slow read enable 0: self-loading, read embedded Flash instruction, read feature control bit instruction, read trim command, the embedded Flash interface adopts normal read mode 1: All embedded Flash Read operations are performed in slow read mode
msdboot_en	[30]	0	Master self-loading dual-boot enable, high effective
reserve	[31]	-	

4. Configuration mode description

This chapter mainly descriptions the application interface and independent development process of the five configuration modes supported by CPLD devices.

(1) JTAG configuration mode

JTAG interface is shown in the figure below:

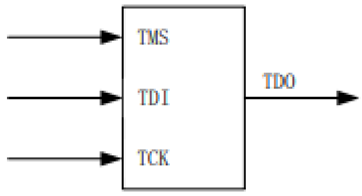


Figure 4 JTAG interface diagram

The list of JTAG ports is as follows:

Table 6 List of JTAG ports

NAME	I/O	DEDICATED / MULTIPLEXED	DESCRIPTION
TCK	I	dedicated	Test clock. TCK provides the clock for chip test logic and is a dedicated input pin. It can be independent from the system clock of the chip, and can also be synchronized with the clock of the working mode.
TMS	I	dedicated	Test mode selection. On the rising edge of TCK, control the state switching of the state machine of the test access port controller to realize the movement of test instructions and test data.
TDI	I	dedicated	Test data entry. Serial input pin. On the rising edge of TCK, the test instruction is shifted into the instruction register and the test data is shifted into the test data register.
TDO	O	dedicated	Test data output. Serial output pin. In the instruction shift state, the test instruction is shifted out of the instruction register on the falling edge of TCK. In the data shift state, on the falling edge of TCK, the test data is shifted out from the test data register placed on the TDI to TDO scan link.

The interface timing is shown in the figure below:

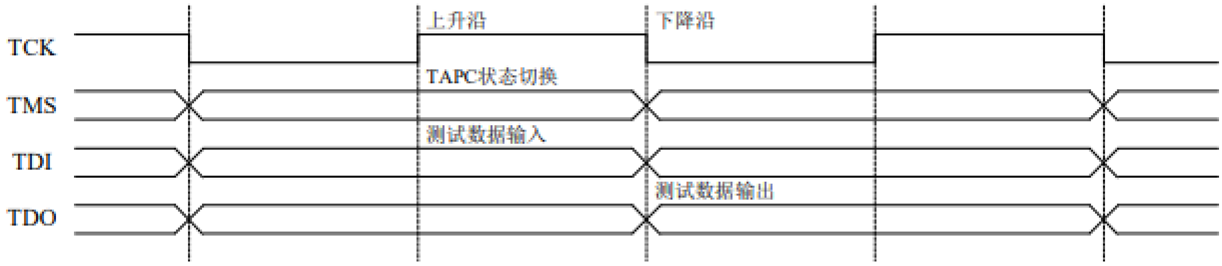


Figure 5 JTAG interface timing

The schematic diagram of the JTAG configuration mode application interface is shown in the figure below.

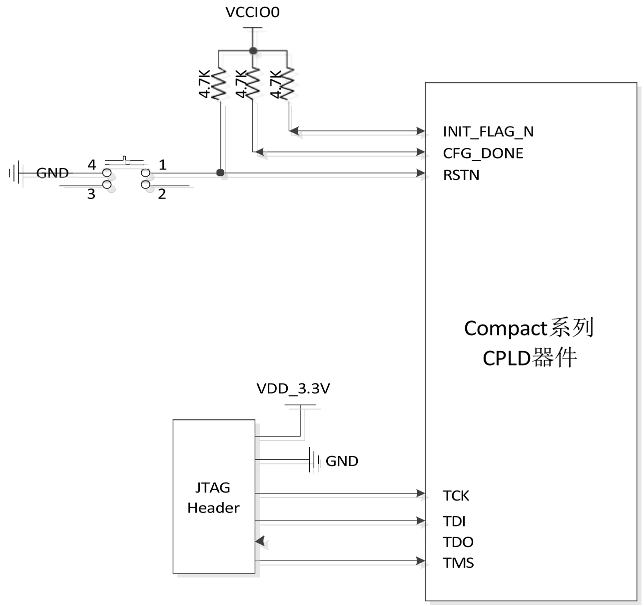


Figure 6 Application diagram of JTAG mode

A low level of RSTN will reset the configuration logic, it is recommended to use a 4.7K resistor to pull up to VCCIO0 externally. During the initialization process, when INIT_FLAG_N is used as an input, if it is connected to a low level, the CPLD device will stay in the initialization stage. It is recommended to use an external 4.7K resistor to pull it up to VCCIO0. When INIT_FLAG_N is used as an output, output high level to indicate the end of chip initialization. CFG_DONE outputs a high level to indicate that the chip enters user mode, and an external input with a low level will make the CPLD device stay in the configuration stage. It is recommended to use an external 4.7K resistor to pull it up to VCCIO0.

In JTAG mode, the TCK test clock needs to be provided externally; the external can control the transition of the JTAG internal TAP state machine by changing the state of TMS, so as to select the writing of configuration bit stream (TDI) or on-chip data readback (TDO) . In addition to configuration programming, the JTAG interface is commonly used for in-circuit debugging and boundary-scan testing.

JTAG cascade mode

Multiple devices can be configured using a JTAG daisy chain, connected as shown in the figure below.

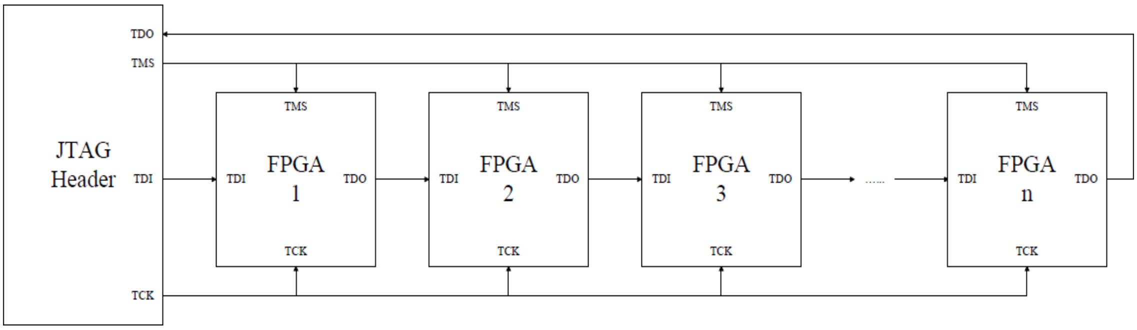


Figure 7 Schematic diagram of JTAG cascade application

In this way, after the download cable is connected, the PDS software on the host will scan all the devices on the JTAG chain, and the user can choose to program and download the corresponding CPLD devices.

The TCK and TMS signals connect all devices in the JTAG chain, so the quality of these two signals will affect the maximum frequency and reliability of the JTAG configuration.

(2) Master SPI configuration mode

When the X1 bit width is applied, the pin MOSI_SI of the CPLD device is connected to the data input terminal of the SPI Flash as a command output, and the pin MISO_SO is connected to the data output terminal of the SPI Flash as a data input.

The schematic diagram of the main SPI configuration mode application interface is shown in the figure below.

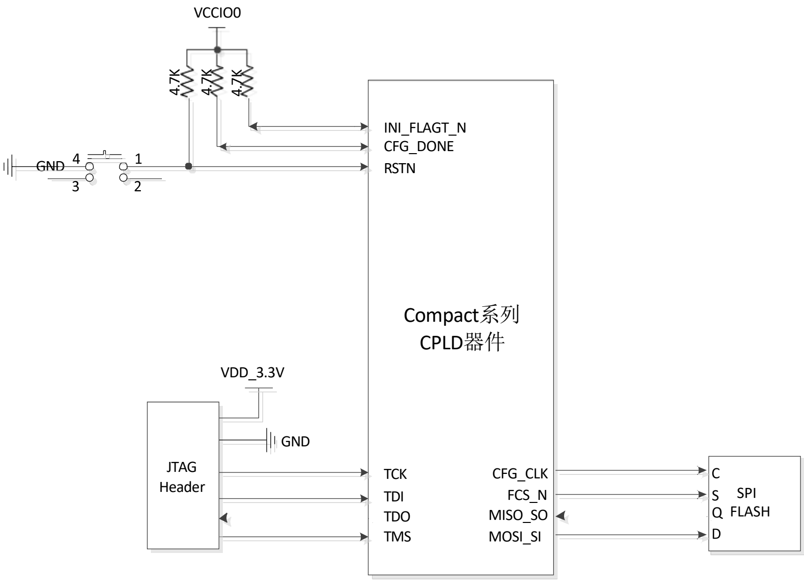


Figure 8 Master SPI Mode Application Diagram

The master SPI mode interface timing is shown in the following figure:

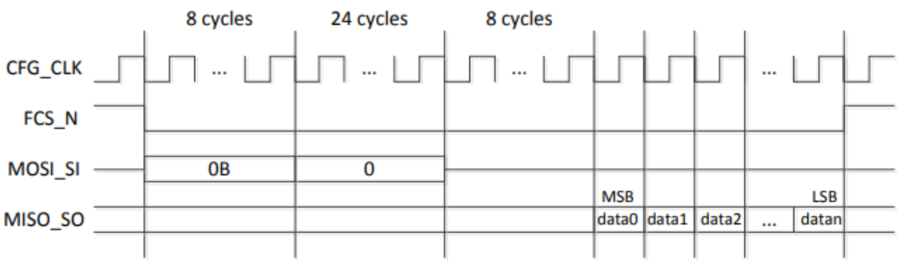


Figure 9 Master SPI mode interface timing diagram

After the main SPI mode is selected, CCS automatically uses the fast read X1 mode to read data from address 0 of the SPI Flash.

The CPLD device generates the SPI Flash chip select signal on the falling edge of CFG_CLK, and sends the operation code and address. SPI Flash samples the opcode and address on the rising edge of CFG_CLK.

SPI Flash generates data on the falling edge of CFG_CLK, and the CPLD device samples data on the rising or falling edge of CFG_CLK according to the setting. When the clock frequency is not higher than 25M, use the rising edge of CFG_CLK to sample data, and when the clock frequency is higher than 25M, use the falling edge of CFG_CLK to sample data.

After all the specified bit streams are fetched from the SPI Flash, FCS_N is set to 1, and the main SPI mode operation ends.

The CPLD device supports direct connection with the 4 pin SPI Flash standard interface, and the specific models supported are shown in the table below.

Table 7 Supported SPIs Flash model

COMPANY	MODEL	CAPACITY	BIT WIDTH	FREQUENCY (MHZ)	DATA DELAY (NS)	OPERATING VOLTAGE (V)	OPERATING TEMPERATURE (CELSIUS)
Micron	M25P05A	512K	X1	50	8	2.3-3.6	-40-85
Micron	M25P10A	1M	X1	50	8	2.3-3.6	-40- 125
Micron	M25P20	2M	X1	75	6	2.3-3.6	-40- 125
Micron	M25P40	4M	X1	75	6	2.3-3.6	-40- 125
Micron	M25PE10	1M	X1	75	6	2.7-3.6	-40- 125
Micron	M25PE20	2M	X1	75	6	2.7-3.6	-40- 125
Micron	M25PE40	4M	X1	75	6	2.7-3.6	-40-85
Micron	M45PE10	1M	X1	75	8	2.7-3.6	-40-85
Micron	M45PE20	2M	X1	75	8	2.7-3.6	-40-85
Micron	M45PE40	4M	X1	75	8	2.7-3.6	-40-85
Winbond	W25Q20CL	2M	X1/X2/X4	104	8	2.3-3.6	-40-85
Winbond	W25Q40CL	4M	X1/X2/X4	104	8	2.3-3.6	-40-85
Winbond	W25X05CL	512K	X1/X2	104	8	2.3-3.6	-40-85
Winbond	W25X10CL	1M	X1/X2	104	8	2.3-3.6	-40-85
Winbond	W25X20CL	2M	X1/X2	104	8	2.3-3.6	-40-85
Winbond	W25X40CL	4M	X1/X2	104	8	2.3-3.6	-40-85
Winbond	W25X40CV	4M	X1/X2	80	8	2.6-3.6	-40- 105
Winbond	W25Q10EW	1M	X1/X2/X4	104	6	1.65-1.95	-40-85
Winbond	W25Q20EW	2M	X1/X2/X4	104	6	1.65-1.95	-40-85

Winbond	W25Q40EW	4M	X1/X2/X4	104	6	1.65-1.95	-40-85
Winbond	W25Q20BW	2M	X1/X2/X4	80	7	1.65-1.95	-40-85
Winbond	W25Q40BW	4M	X1/X2/X4	80	7	1.65-1.95	-40-85
Spansion	S25FL204K	4M	X1/X2	85	10	2.7-3.6	-40-85
Zhao Yi	GD25D05B	512K	X1/X2	80	6	2.7-3.6	-40-85
Zhao Yi	GD25D10B	1M	X1/X2	80	6	2.7-3.6	-40-85
Zhao Yi	GD25Q20C	2M	X1/X2/X4	120	7	2.7-3.6	-40-85
Zhao Yi	GD25Q40C	4M	X1/X2/X4	120	7	2.7-3.6	-40- 125
Zhao Yi	GD25Q21B	2M	X1/X2/X4	104	6	2.7-3.6	-40-85
Zhao Yi	GD25Q41B	4M	X1/X2/X4	104	6	2.7-3.6	-40-85
Zhao Yi	GD25VQ20C	2M	X1/X2/X4	104	7	2.3-3.6	-40-85
Zhao Yi	GD25VQ40C	4M	X1/X2/X4	104	7	2.3-3.6	-40-85
Zhao Yi	GD25VQ21B	2M	X1/X2/X4	104	6	2.3-3.6	-40-85
Zhao Yi	GD25LQ05B	512K	X1/X2/X4	104	7	1.65-2.1	-40-85
Zhao Yi	GD25LQ10B	1M	X1/X2/X4	104	7	1.65-2.1	-40-85
Zhao Yi	GD25LQ20B	2M	X1/X2/X4	104	7	1.65-2.1	-40-85
Zhao Yi	GD25LQ40B	4M	X1/X2/X4	104	7	1.65-2.1	-40-85
Zhao Yi	GD25LQ40	4M	X1/X2/X4	120	7	1.65-1.95	-40-85

Note: The models in the table are currently tested Flash, if you use other models, you need to test and verify.

(3) Master self-loading configuration mode

The master self-loading configuration mode is designed for embedded Flash. First set the feature control bit, then load the bit stream into the embedded Flash, reset the device, and CCS will automatically read the bit stream from the embedded Flash to configure the CPLD.

(4) From SPI configuration mode

Pango Compact series CPLDs provide the slave SPI interface as one of the access interfaces of the chip CCS. Through the SPI interface, users can configure/reconfigure CRAM, program embedded FLASH, read and write feature control bits, etc.

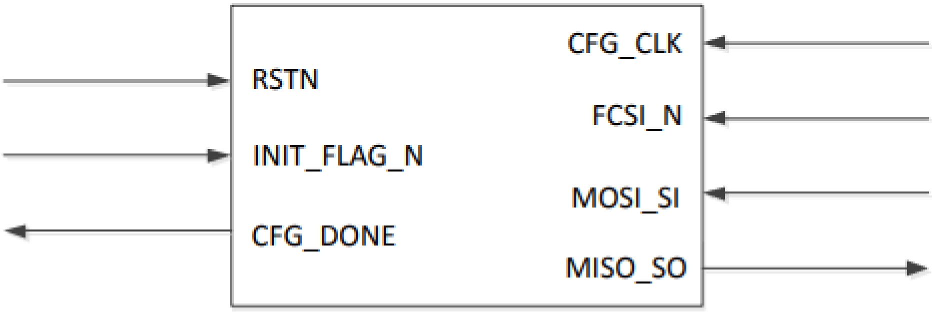


Figure 10 Slave SPI Interface Diagram

The port list is shown in the table below

Table 8 From SPI port list

NAME	I/O	DESCRIPTION
RSTN	I	Reset pin, active low. Asynchronous reset of full chip except JTAG TAP controller. When it remains low, the chip cannot be configured. INIT_FLAG_N is low at the same time. Low level triggers reset, the chip configuration returns to the initial state, restarts to clear the configuration storage, and restarts the configuration process. When the pin is floating, it is recommended to make the pin a fixed weak pull-up state
CFG_CLK	I	Input clock, up to 100M. When executing READ and READ_CTL instructions, the time to transfer stuff bytes cannot be less than 800ns. This can be achieved in a number of ways: 1. Constant frequency clock. When executing READ and READ_CTL instructions, the clock frequency is up to 10M. 2. The highest clock frequency is 100M. Clock discontinuity during transfer of stuff bytes. After transmitting the first half of stuffing bytes, the clock pauses for no less than 800ns, and then continues to transmit the second half of stuffing bytes. After the CPLD device enters the user mode, if the SPI interface continues to be used as a configuration pin, the pin will continue to be used as a configuration pin, and if the SPI interface is released to the user, the pin will be released to the user.

INIT_FLAG_N	OPEN DRAIN	CPLD device initialization completion flag signal; configuration error flag signal After the chip is powered on or reset, this pin is a bidirectional port, driven by the internal circuit to low level, and input to the chip at the same time; when the CPLD device is initialized, this pin is high (external pull-up), when the pin is high, the mode pin is sampled. This pin is set as an open-drain output and is pulled high by an external pull-up resistor. During the configuration process, the pin is bidirectional, and when an error occurs, the pin is driven by the internal circuit, and the output is low, and the state of the pin is fed back to the chip. After the initialization of the CPLD device is completed, the pin can continue to be driven low by the outside to achieve the purpose of delaying the configuration until the pin changes from low to high (must be set as an open-drain output). After the CPLD device enters the user mode, this pin can continue to be used as a configuration pin, or it can be released to the user.
CFG_DONE	OPEN DRAIN	Indicates that the configuration is complete. 0 : CPLD Device not configured 1 : CPLD Device configured
FCSI_N	I	Chip select input pin of SPI interface, active low. After the CPLD device enters the user mode, if the SPI interface continues to be used as a configuration pin, the pin will continue to be used as a configuration pin, and if the SPI interface is released to the user, the pin will be released to the user.
MOSI_SI	I	SPI data bus master output slave input pin. After the CPLD device enters the user mode, if both the master SPI interface and the slave SPI interface are released to the user, the pin MOSI_SI is released to the user.
MISO_SO	O	SPI data bus master input slave output pin. After the CPLD device enters the user mode, if both the master SPI interface and the slave SPI interface are released to the user, the pin MISO_SO is released to the user.

Provides a convenient and versatile way for user configuration from the SPI interface. Compared with the JTAG interface, it is easier to implement, and has a higher transmission rate than the slave I2C interface. Compared with the master self-loading and master SPI modes, it has more flexible version management capabilities and richer chip operations. Usually, the CPU, MCU, DSP and other devices can be used as the master device, and the CPLD can be used as the slave device. This application method can be represented by the following figure:

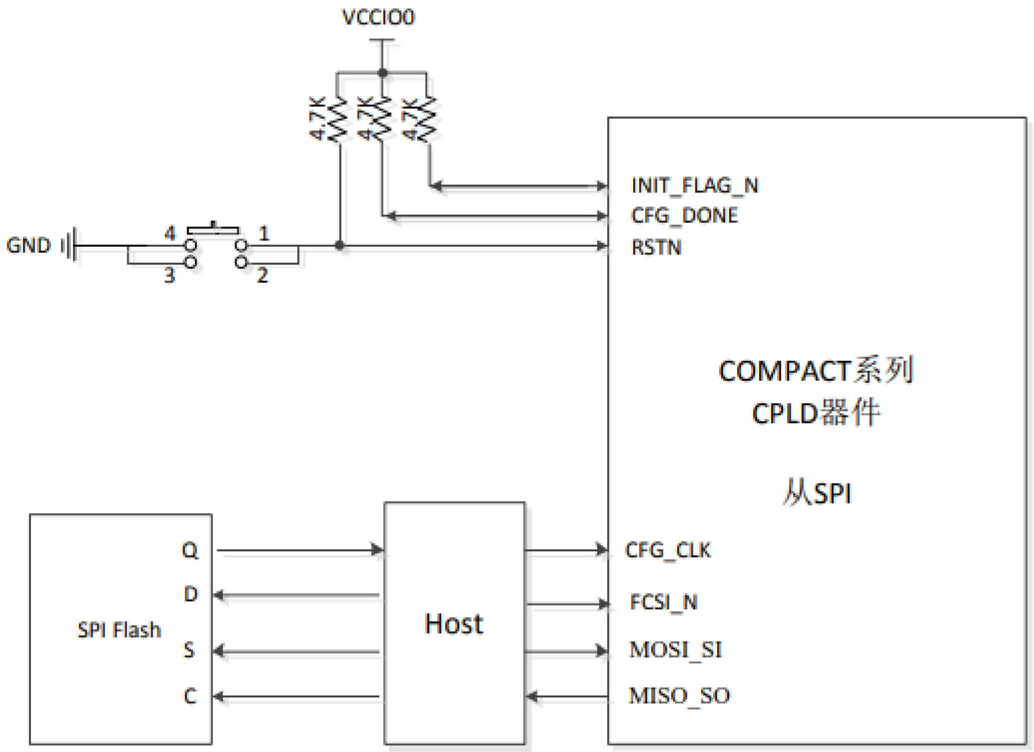


Figure 11 Schematic diagram of slave SPI application

Pango provides the implementation of the C code from the SPI operation process, please refer to the application guide "AN03009 Compact Series CPLD Device Configuration and Programming Application Guide from the SPI Interface", and the corresponding sample project.

For the introduction of using the SPI interface, refer to Chapter 7 for configuration details. Special attention should be paid to the fact that FCSI_N needs to be pulled high before the start of all instructions and after the instruction is completed, and at least 8 clocks should be given.

(5) Slave I²C configuration mode

Pango Compact series CPLDs provide the I²C interface as one of the access interfaces of the chip CCS. The slave I²C mode interface supports fast mode. Through the slave SPI interface, users can configure/reconfigure CRAM, program embedded FLASH, read and write feature control bits, etc.

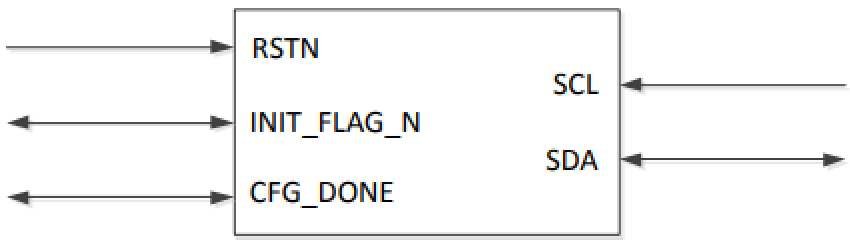


Figure 12 Slave I²C interface diagram

The list of ports from the I²C interface is as follows:

Table 9 Slave I²C port list

NAME	I/O	DESCRIPTION
SCL	I	Serial clock, the highest frequency is 400KHz
SDA	IO	Serial data

Occupying less IO from the I²C interface has the advantage of saving hardware resources. Usually, the CPU,

MCU, DSP and other devices can be used as the master device, and the CPLD is used as the slave device. This application method can be represented by the following figure:

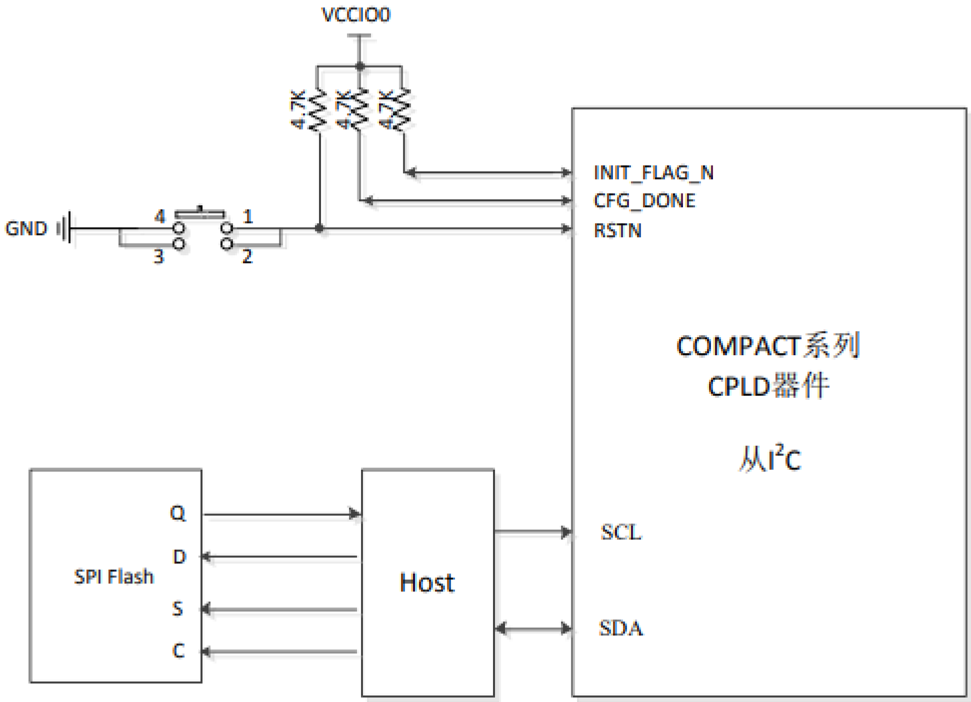


Figure 13 Schematic from I²C application

Pango provides the implementation of the C code from the I²C operation process, please refer to the application guide "AN03010 Compact Series CPLD Device Configuration and Programming Application Guide from the I²C Interface", and the corresponding sample project.

Note that if you want to still operate the device via slave I²C in user mode, you should set the I²C reserved bit of the feature control bit, bit[16]persist_si2c_n, to 0. And cancel the automatic shutdown of OSC, corresponding to the PDS setting Project Setting->Generate Bitstream->Configuration->Enable OSC Shut Off Uncheck.

5. Dual boot

All CPLD devices support dual-boot function, that is, two sets of bit streams are stored in the embedded Flash and the external SPI Flash respectively.

When dual-booting, you need to turn on the watchdog (check the configuration process and timeout in user mode, and the watchdog count will be reduced by 1 every 512 system clock cycles). If there is an error when downloading the current application bit stream, reset the circuit other than the version fallback circuit, and the device can download the golden bit stream again; if there is an error in the configuration of the golden bit stream, then it will not be reset, but INIT_FLAG_N is set to 0, dual-boot The operation ends and the watchdog is disabled.

When the device uses master self-loading, the application bit stream is stored in the embedded Flash, and the golden bit stream is stored in the external SPI Flash; when the device uses the main SPI loading, the application bit stream is stored in the external SPI Flash, and the golden bit stream is stored in the embedded Flash.

To judge whether there is an error in the configuration, you can read the timeout, crc_err and id_err of the status register to judge. In addition, the fallback of the status register indicates whether the current bit stream is the bit stream that falls back after an error.

Only D-type devices support the dual-boot function of master and self-loading, which is characterized in that two sets of bit streams can be stored in the embedded Flash: gold bit stream and application bit stream, and no external SPI Flash is required.

The starting address of the golden bit stream is 0, the application bit stream after the golden bit stream, and the user Flash after the application bit stream.

The watchdog needs to be turned on when the main self-loading dual boot. The flow of the main self-loading dual boot is as follows.

1. Load the golden bitstream from the embedded Flash 0 address.

2. The master-self-load dual-boot command in the golden bitstream loads the application bitstream from the embedded Flash address specified by DBOOTADRR.

3. If there is an error in the process of loading the application bit stream, reset the circuit other than the version fallback circuit, and load the gold bit stream from the embedded Flash 0 address.

4. If there is an error in the gold bitstream configuration process, it will not be reset, but INIT_FLAG_N will be set to 0, and the dual-boot operation of the master and self-loading will end.

5. After the application bit stream or gold bit stream is configured successfully, enter the user mode.

6. In user mode, the application bit stream and user Flash content can be updated internally from the APB interface, JTAG interface, from the SPI interface or from the I2C interface.

7. After the application bit stream and user Flash content are updated, it can be upgraded through internal reset from the APB interface, JTAG interface, SPI interface or I2C interface.

Operating procedures

For non-master self-loading dual-boot, just turn on the dual-boot enable, that is, set bit[9]fallback_en of the feature control bit to 1.

The bit stream in embedded Flash and external SPI Flash should be consistent with the bit stream in master self-loading and master SPI mode.

For the main self-loading dual-boot, it is necessary to enable the dual-boot enable and the main-self-loading dual-boot enable at the same time, that is, set bit [9] fallback_en and bit [30] msdboot_en of the feature control bit to 1.

The main self-loading dual-boot bitstream can be generated with the PDS download tool. Firstly, two sfc files need to be generated, which are golden bit stream and application bit stream respectively. The Flash device settings remain default, as shown in the following figure:

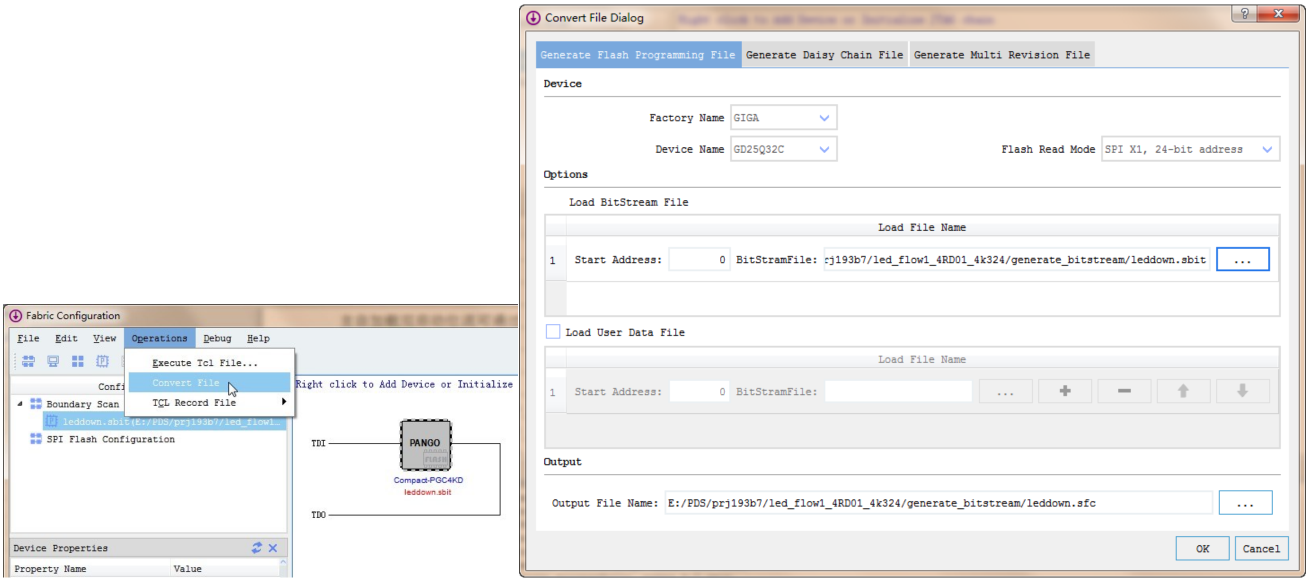


Figure 14 Generate golden bitstream or apply bitstream sfc

Then continue to synthesize the main self-loading dual-boot bitstream through the conversion tool, and the bitstream at the first bitstream position is used as the golden bitstream. As shown below:

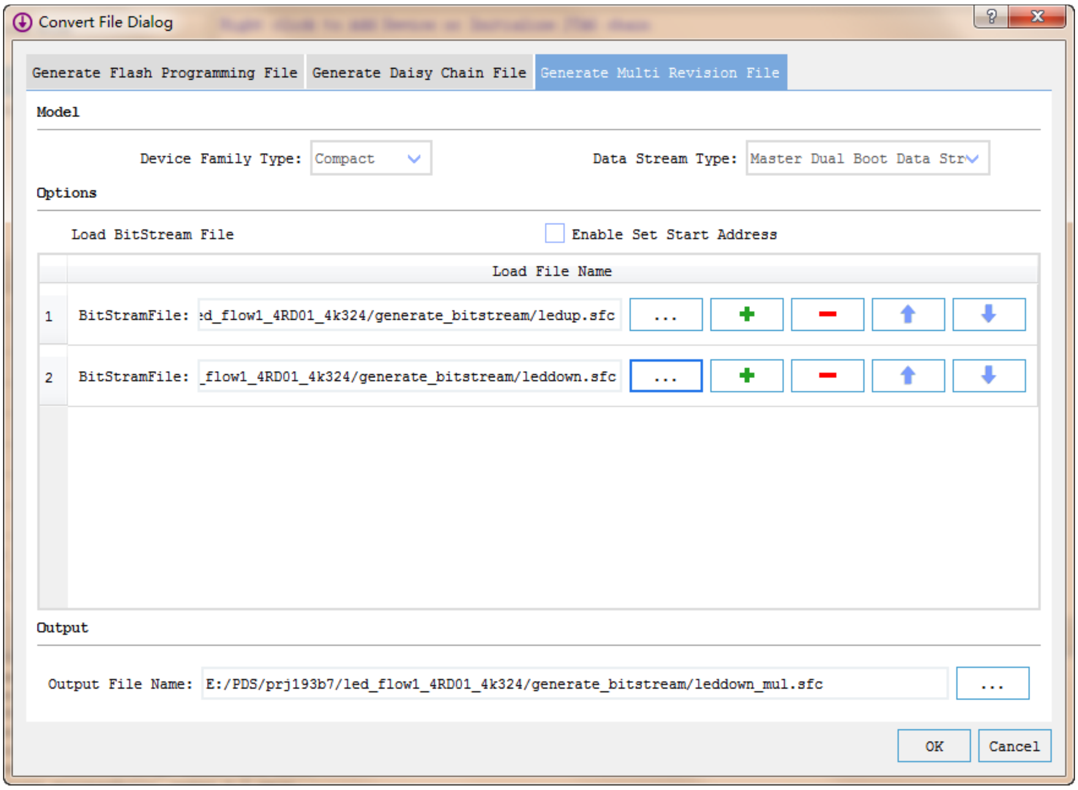


Figure 15 Generate main self-loading dual boot bitstream

By default, the bit stream sbit file that configures the CPLD is used when programming the embedded Flash. The generated file is an sfc file. If you want to download it to the embedded Flash, you need to modify the file used to program the embedded Flash in setting device properties. As shown below:

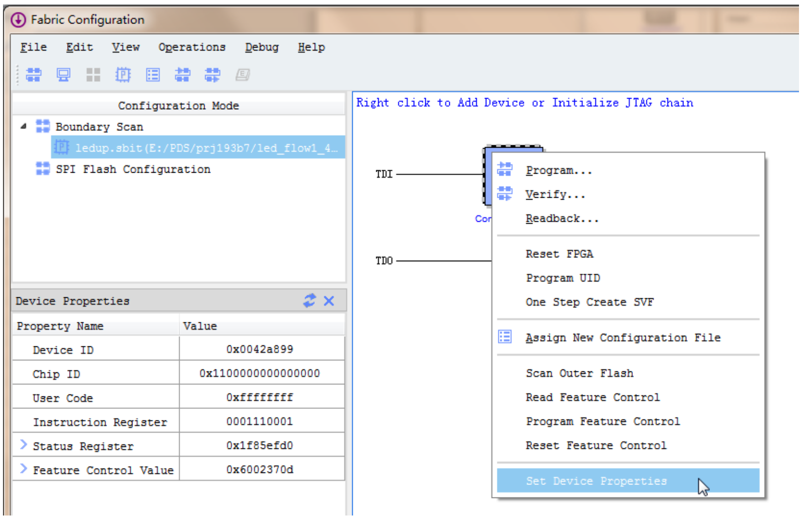


Figure 16 Modify device properties

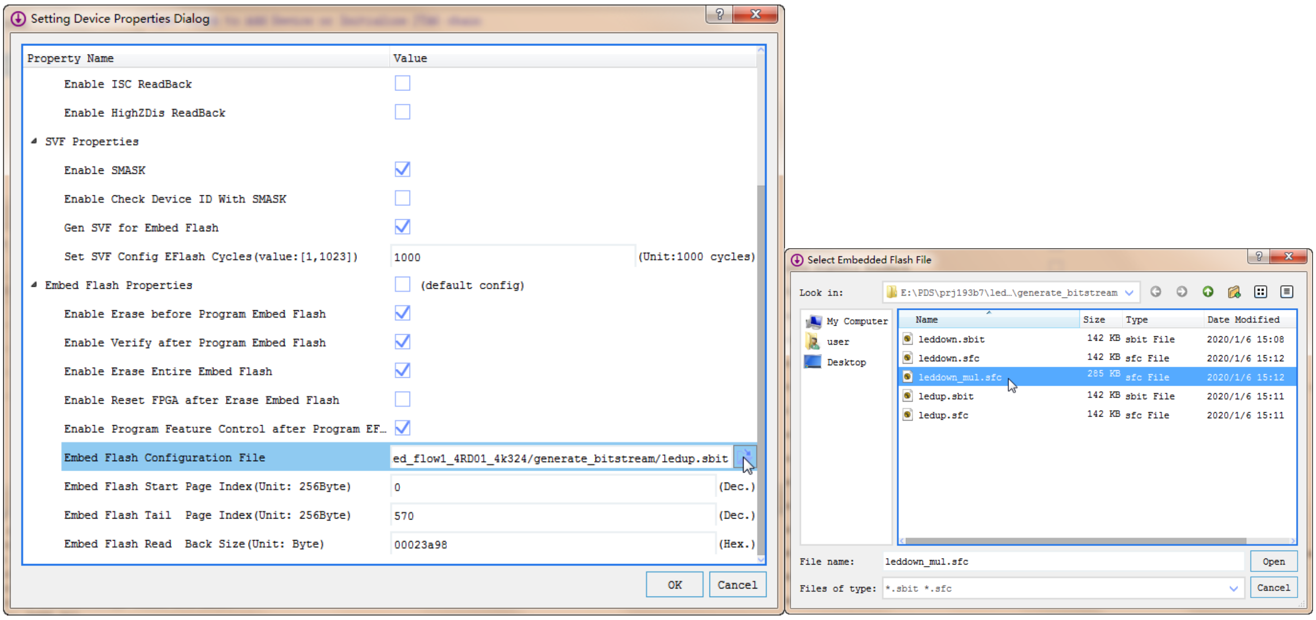


Figure 17 Select the main self-loading dual-boot bitstream sfc file

Then program to embedded Flash. For the specific operation of updating the application bit stream and user data in user mode, please refer to Chapter 7 Configuration Details and Chapter 8 Boundary Scan and JTAG Configuration.

After updating the embedded Flash, send a reset command to upgrade the bit stream in the configuration memory. Reset methods include internal APB interface reset, JTAG interface reset, SPI interface reset, I^2C interface reset, and RSTN pin hard reset (the feature control bit needs to be set to reserve the RSTN pin).

6. PDS Software download configuration

The CPLD device has integrated the related operations of JTAG and main SPI configuration in the PDS operating software. For the required configuration files, the PDS design software will generate configuration files with different extensions to adapt to different configuration schemes. See the table below for specific descriptions.

Table 10 PDS configuration file description

CONFIGURATION FILE EXTENSION	DESCRIPTION
.sbit	Binary configuration data containing header information (bitstream name, date, etc.) that is recognized by configuration tools but not written to the CPLD device. Use the Fabric Configuration tool to directly write the .sbit file to the CPLD device through the Cable
.bin	Binary configuration data (pure bit stream) without header information, suitable for user configuration schemes, such as microprocessor configuration of CPLD devices.
.sfc	Bitstream file written to Flash, converted from .sbit file

The .sbit file is the default bitstream file generated after the PDS tool is compiled. Run Generate Bitstream to generate the .sbit file. For the specific operation process, please refer to "Design Editor User Guide" and "Fabric Configuration User Guide".

CPLD devices support normal and compressed bit streams. The compressed bit stream file is smaller than before compression, and the size of the compressed file is related to the design.

Table 11 The bit stream size of each device in the CPLD device family

COMPACT SERIES CPLD DEVICE	FILE NAME	UNCOMPRESSED FILE SIZE (KB)
1KL	*.sbit	53
1KG	*.sbit	83
2KL	*.sbit	83
2KG	*.sbit	83
4KL	*.sbit	142
4KD	*.sbit	142
7KD	*.sbit	222

7. Configuration details

(1) Package format

The bit stream consists of multiple packets, and the packet consists of a header part and a data part.

The header is a 32-bit word. After the header is the data. Data is in units of 32-bit words.

If the number of 32-bit words following the header specified by the header part is 0, then this packet has only the header part and no data part, and is an empty packet. The package format is as follows:

HEADER TYPE [31:29]	OPCODE [28:27]	REGISTER ADDRESS [26:22]	32-BIT WORD COUNT [21:0]
101	00: no operation 01: write 10: read 11: reserved	register address	The number of 32-bit words following the packet header

(2) Configuration registers

The CCS system of CPLD devices include the following registers. When writing the register address, write the high bit first.

Table 12 Configuration register address list

NAME	R/W	ADDRESS	DESCRIPTION
CRCR	R/W	00000	CRC register
CMDR	R/W	00010	Command register
CTRLR	R/W	00100	Control register
MFWRITER	W	00110	Multi-frame write register
STATUSR	R	01001	Status register
DBOOTADRR	R/W	10000	Dual Boot Address Register
WATCHDOGR	R/W	10001	Watchdog register
CMASKR	R/W	10111	Control mask register
RBCRCR	R/W	11110	Readback CRC register

1. CRC Register (CRCR)

Each write to the CRC register corresponds to a CRC check of the bit stream. The CRC check passes if the value written to the CRC register is the same as the current CRC value calculated by the CCS. Otherwise INIT_FLAG_N (init_complete) is pulled low.

2. Command Register (CMDR)

The command register determines the CCS What to do next.

Table 13 Command Register Instruction Description

COMMAND	CODING	DESCRIPTION
NOP	00000	no operation (initial value)
RSTCRC	00001	Reset CRC register
WCMEM	00100	Write configuration data, used before writing configuration memory configuration data via CMEMIR
MFWRITE	00101	Multi-frame write, write the currently written frame of data into the next continuous frame. The number of frames is determined by MFWRITER
RCMEM	00110	read configuration data, use the CMEMOR Before reading back configuration memory configuration data
SWAKEUP	00111	start wake up
SWAKEDOWN	01000	Turn off wakeup action
GUP	01001	Internal logic enable
GDOWN	01010	Internal logic is not enabled
DESYNC	01011	Desync, used at the end of configuration
RWD	01100	Restart watchdog
WCMEMEN	10000	Write configuration memory enable, used to enable configuration memory write before writing configuration data
WCMEMDIS	10001	Write configuration memory prohibition, after the configuration data is written, the configuration memory is prohibited from writing
RCMEMEN	10010	Read configuration memory enable, used to enable configuration memory read before reading configuration data
RCMEMDIS	10011	Read configuration memory prohibition, after the configuration data is read, the configuration memory is prohibited from being read
MSDBOOT	10101	Master self-loading dual-boot, starting to load the application bitstream from the address specified by the dual-boot address register

3. Control Register (CTRLR)

Table 14 Control Register Description

NAME	BIT	INITIAL VALUE	DESCRIPTION
reserve	[31:5]		
mask_en	[4]	1'b0	Variable memory location readback mask enable 0 : disable masking 1 : enable shielding 2K The device does not have this function
mfg_sram_retention	[3]	1'b0	When it is 1 (and gwen=1), the SRAM retention voltage test is turned on (Domain is 1.2V)
osc_off_en	[2]	1'b1	Controls whether the configuration control system logic is allowed to shut down the OSC 0 : not allowed 1 : allow
wrctrl	[1:0]	2'b00	External port security level. Control shutdown reconfiguration, partial reconfiguration, partial dynamic reconfiguration, and readback configuration memory. Once configured as disabled, it cannot be changed to enabled unless reset 00: reconfiguration enabled, readback enabled (default) 01: Reconfiguration enabled, readback disabled 1x: reconfiguration disabled, readback disabled

4. Multi-Frame Write Register (MFWRITER)

The number of compressed frames means that the data content of the following consecutive frames is the same as the data content of the just written frame. For example, after configuring a frame of data , if the data content of the following three consecutive frames is the same as this frame, then the multi-frame write register content needs to be written as 3 .

5. Status Register (STATUSR)

The status of the configuration pins of the chip and the working status of the chip are recorded in the status register, which can be read from the SPI or I^{2}C interface through JTAG. Details are shown in the table below.

Table 15 Status Register (STATUSR)

NAME	BIT	DESCRIPTION
reserve	[31:30]	
persist_mspi	[29]	SPI in user mode Interface enable, high effective

persist_si2c	[28]	User Mode from I^{2}C Interface enable, high effective
persist_sspi	[27]	User mode from SPI Interface enable, high effective
persist_jtag	[26]	JTAG in user mode Interface enable, high effective
persist_done	[25]	CFG_DONE in user mode Pin enable, active high
persist_init	[24]	INIT_FLAG_N in user mode Pin enable, active high
persist_rstn	[23]	RSTN in user mode Pin enable, active high
pass_cal	[22]	Embedded FLASH Calibration successful flag, high effective 4K/7K device this bit is reserved
busy	[21]	Embedded Flash Busy flag, active high
lock	[20]	Embedded Flash Lock flag, active high Embedded Flash When locked, no programming, no reading of bit stream data and user Flash , can only erase the whole Flash
wake up	[19]	Embedded Flash Wake-up flag, active high
sleep	[18]	Embedded Flash Sleep flag, active high
fallback	[17]	Fallback indicator, active high
pll_lock	[16]	PLLs Lock flag, active high
gwen	[15]	Global write enable, active high
grs_n	[14]	Global register set reset, active high
go out	[13]	Global I/O Output enable, active high
reserve	[12]	
glogen_fb	[11]	Global Logic Enable Feedback
glogen	[10]	Global logic enable, active high
done_i	[9]	DONE pin input
done	[8]	Device wake-up success flag, active high
init_n	[7]	INIT_FLAG_N pin input
init_complete	[6]	Initialization complete and configuration error indication, active high
wakedown_over	[5]	Wake-up shutdown is over, active high
wakeup_over	[4]	Wake-up is over, active high
timeout	[3]	Watchdog timeout, active high
rbcrc_err	[2]	Readback CRC Test results 0: CRC Correct

		1: CRC wrong
crc_err	[1]	CRC Test results 0: CRC Correct 1: CRC wrong
id_err	[0]	ID Test results 0: Correct 1: wrong

6. Dual Boot Address Register (DBOOTADRR)

Table 16 Dual Boot Address Register Description

NAME	BIT	DESCRIPTION
reserve	[31:19]	
Ba[1:0]	[18:17]	heap address
Ra[8:0]	[16:8]	page address
reserve	[7:6]	
Ca[5:0]	[5:0]	32 Bit data page offset address

for 1K devices, Ba[1:0] are reserved bits.

for 2K device, Ba[1:0] is the reserved heap address and needs to be set to 2'b11 .

7. Watchdog Register (WATCHDOG)

Used for timeout detection in configuration mode.

Table 17 Watchdog Register Description

NAME	BIT	INITIAL VALUE	DESCRIPTION
wd_user_en	[31]	1'b0	Watchdog enable in user mode for shutdown operation
wd_cfg_en	[30]	1'b0	Watchdog enable in configuration mode
wd_value	[29:0]	30'h3FFF_FFFF	Watchdog timeout value.

8. Control Mask Register (CMASKR)

The corresponding bit in the mask control register, the default value is 32’d0. 0 is mask. For example, when the value of the control mask register is 32’h0000_0003, the lower 2 bits of the control register can write data, and other bits cannot write data.

9. Option Register 0 (OPTION0R)

Table 18 Option Register Description

NAME	BIT	INITIAL VALUE	DESCRIPTION
reserve	[31]		
oscfsel_rbcrc	[30:24]	7'd0	Readback CRC Clock Frequency Selection 7'b0000000 : 266/128, 2.08MHz 7'b0000001 : 266/2, 133.00MHz 7'b0000010 : 266/2, 133.00MHz 7'b0000011 : 266/3, 88.67MHz 7'b0000100 : 266/4, 66.50MHz ... 7'b1111110 : 266/126, 2.11MHz 7'b1111111 : 266/127, 2.09MHz
done_syn	[23]	1'b0	External done synchronization enable. Indicates whether the external input done is a synchronization signal to the output done. If it is a synchronous signal, the wake-up module of CCS uses it directly. If it is an asynchronous signal, it needs to be synchronized before use, usually used to delay the done function. 0: out of sync 1: synchronous
reserve	[22:12]		
wait_pll	[11]	1'b0	Wait for PLL lock enable on wakeup 0 : No need to wait for PLL locking 1 : Need to wait for PLL locking
startup_sel	[10:8]	3'b000	Wake up clock selection 000 : mclk 001 : sclk 010 : tck 011 : scl 100 : uclk 101 : spal_clk The wake-up clock can only choose the clock used during configuration or the user clock
reserve	[7]		
oscfsel	[6:0]	7'd0	Master SPI Clock MCLK frequency selection 7'b0000000 : 266/128, 2.08MHz 7'b0000001 : 266/2, 133.00MHz

			7'b0000010 : 266/2, 133.00MHz 7'b0000011 : 266/3, 88.67MHz 7'b0000100 : 266/4, 66.50MHz ... 7'b1111110 : 266/126, 2.11MHz 7'b1111111 : 266/127, 2.09MHz

10. Readback CRC Register (RBCRCR)

CRC value for non-DRM data in configuration memory.

(3) Slave SPI/slave I²C instruction set

Table 19 Slave SPI/ slave I²C instruction set

INSTRUCTION	DESCRIPTION	OPCODE
NOP	No Operation	FF
RDID	Read Identification	A1
RD USER	Read Usercode	A2
RDSR	Read Status Register	A3
RDUID	Read Unique Identification	A4
RDLOCK	Read Embedded Flash lock Information	A5
CFG	Config Bitstream	50
WREN	Write enable	51
WRDIS	Write Disable	52
RESET	Reset CPLD	60
ERASE	Erase Bulk	10
ERASE_PAGE	Erase Page	11
ERASE_CTL	Erase Features control bit	12
PROGRAM	Program page	20
PROGRAM_CTL	Program Features Control bit	22
READ	Read	30
READ_CTL	Read Features control bit	31
PROGRAM_LOCK	Lock Embedded Flash	40
EFlash_SLEEP	Embedded Flash sleep	70
EFlash_WAKEUP	Embedded Flash wake up up	71

Note: For detailed introduction of slave SPI/slave I²C instruction timing, refer to appendix 4 and appendix 5 respectively. Special attention should be paid to the fact that during the operation from the SPI interface, FCSI_N needs to be pulled high before all instructions start and after the instruction is completed, and at least 8 clocks are given.

(4) Configure CRAM (configuration memory) via SPI/I²C interface

Configuration / reconfiguration

Configuration, CPLD chip configuration from the beginning to the process of entering the user mode; reconfiguration, after the CPLD chip enters the user mode, reset (command reset or external trigger RSTN hard reset), clear all CRAM, exit the user mode, and then reconfigure process.

The process is as follows:

1. Power on

2. After the value of INIT_FLAG_N is 1, write the RDID command and read the device IDCODE

3. Check the value of IDCODE

4. If the IDCODE does not match, terminate the operation. If IDCODE matches, write WREN instruction

5. Write the CFG instruction to load the bit stream

6. After the bit stream is sent, if it is released from the SPI/I²C interface to the user in user mode (the feature control bit persist_sspi_n/persist_si2c_n is 1), check the values of INIT_FLAG_N and CFG_DONE, and the operation ends. If the SPI interface is used as a configuration interface in user mode, write the WRDIS command

7. Write the RDSR instruction and read the device status register. At any time after power-on, the CPLD device can be cold-started (set RSTN to 0). After a cold start, you can go to step 2 to reconfigure the CPLD device.

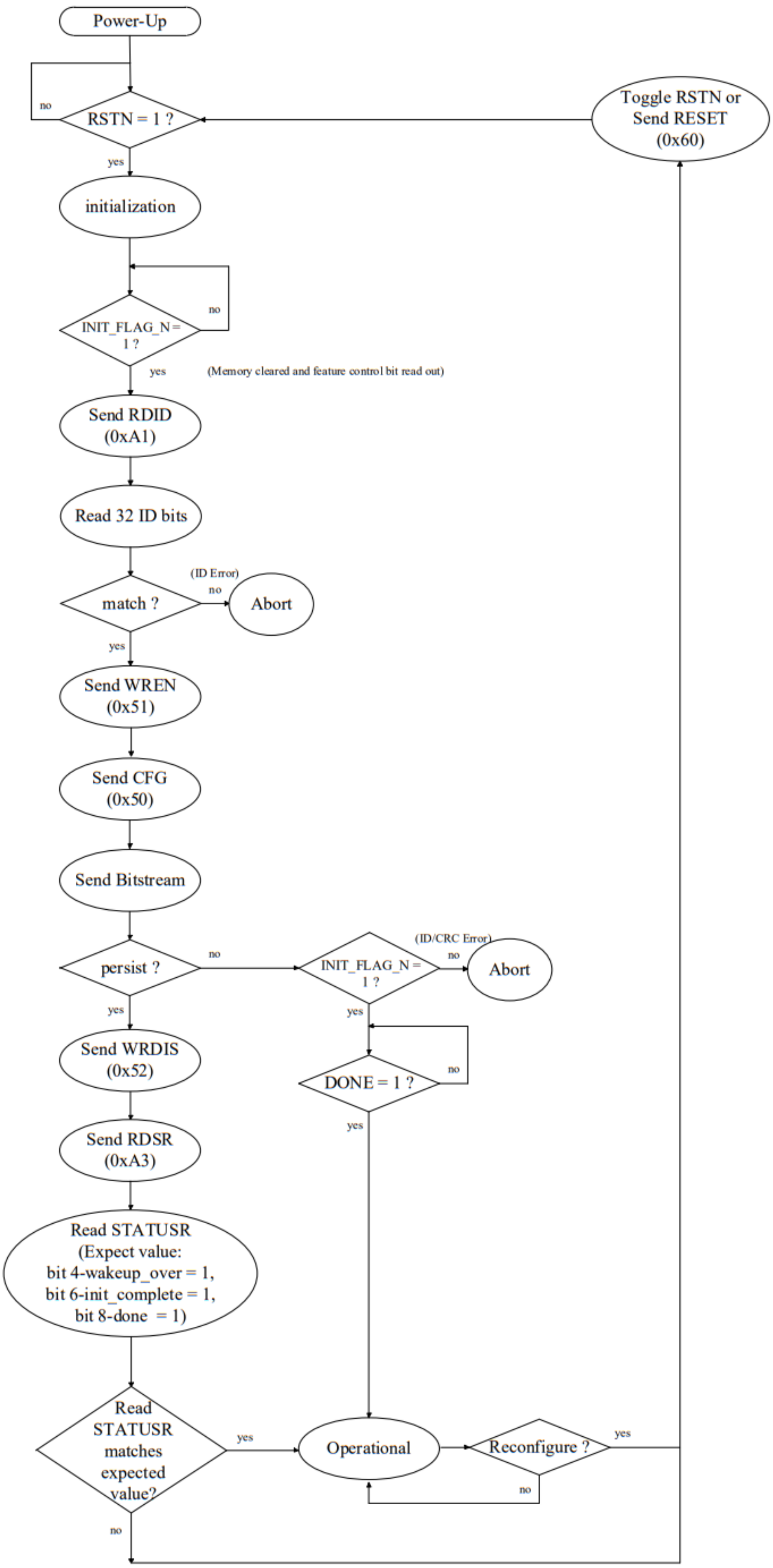


Figure 18 Configuration / Reconfiguration Flowchart

Shutdown reconfiguration

Power-off reconfiguration without resetting, without clearing all CRAM, without exiting user mode, powering down the device for reconfiguration/partial reconfiguration. Power-off reconfiguration needs to be implemented by writing a series of commands to the CCS. These commands complete the modification of the configuration registers in accordance with the bit stream packet format to realize the power-off reconfiguration function.

Table 20 Shutdown reconfiguration from SPI

PROCESS	DATA
Write WREN command	0x51
Write CFG command, load:	0x50
100 filler words	0xFFFFFFFF ... 0xFFFFFFFF
Sync word	0x01332D94
No operation	0xA0000000
Write RSTCRC command to CMDR register	0xA8800001 0x00000001
Write OPTION0R register (select wake-up shutdown clock)	0xAE400001 0x00000100
Write CMASKR register	0xADC00001
Write ctrlr Registers (disable masked variable memory location readback)	0x00000010
Write SWAKEDOWN command to CMDR register	0xA8800001 0x00000008
20 no-op	0xA0000000 ... 0xA0000000
Write GDOWN command to CMDR register	0xA8800001 0x0000000A
Write DESYNC command to CMDR register	0xA8800001 0x0000000B
Bit stream	bit stream
Write WRDIS instruction	0x52
Write RDSR command, read device status register	0xA3

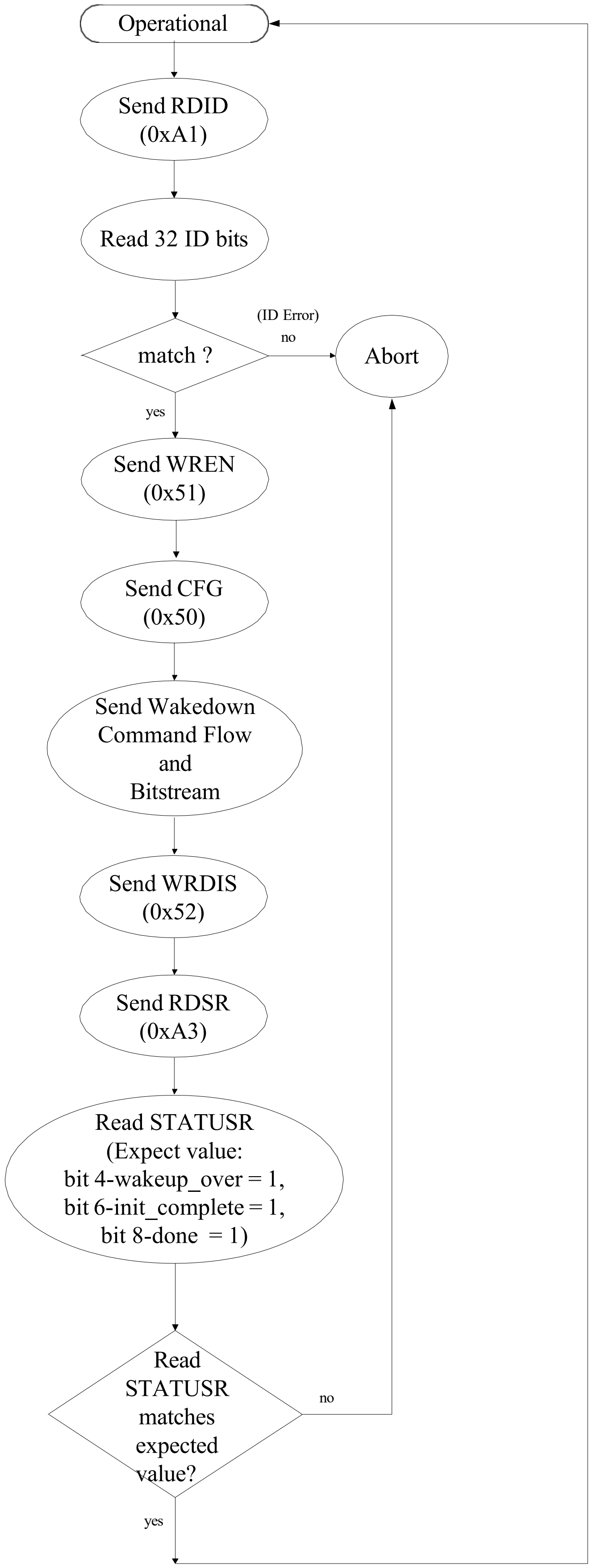


Figure 19 shutdown reconfiguration process

(5) Programming embedded Flash via SPI/I²C interface

Embedded Flash supports direct operation of embedded Flash through JTAG, from SPI, and from I²C interface. The operation process of slave SPI and slave I²C is the same, only the interface timing is different. This section introduces the operation process of slave SPI and slave I²C. For the JTAG operation flow, see Chapter 8 Boundary Scan and JTAG Configuration.

From SPI, the complete process of programming embedded Flash from I²C interface is as follows:

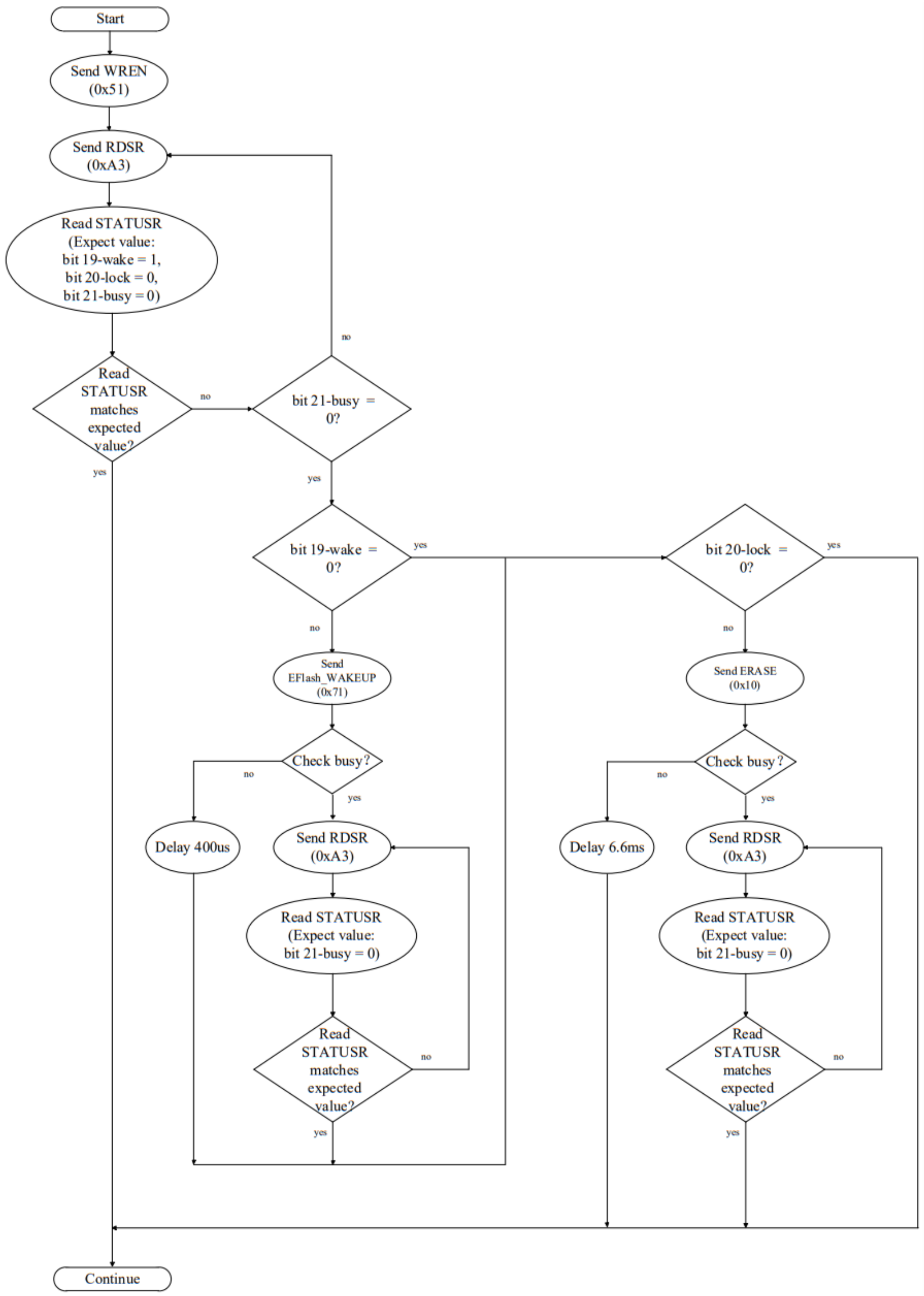


Figure 20 Programming Embedded Flash 1

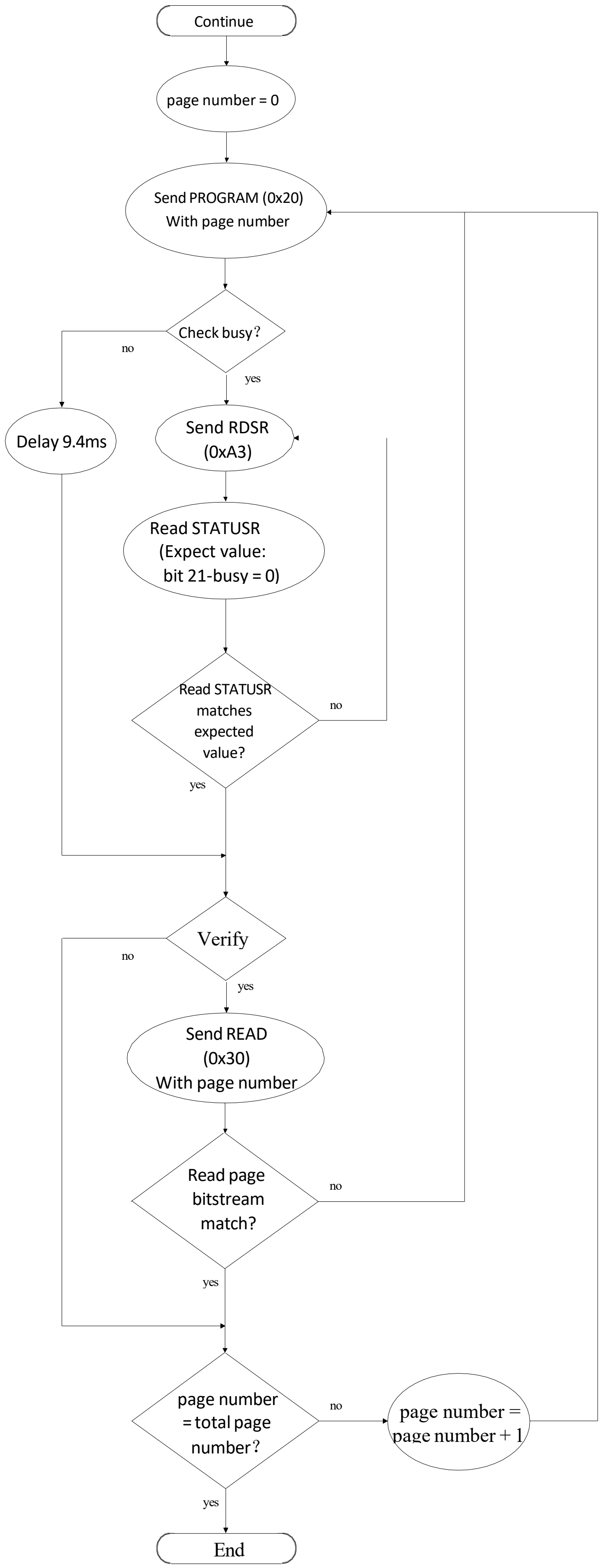


Figure 21 Programming Embedded Flash 2

(6) Internally from the APB Interface Programming Embedded Flash

The Compact series CPLD supports the internal slave APB interface through user logic operation to realize the operation of the embedded Flash. The input of user logic can be JTAG interface or user-defined interface. In-system indirect programming logic block diagram is shown below.

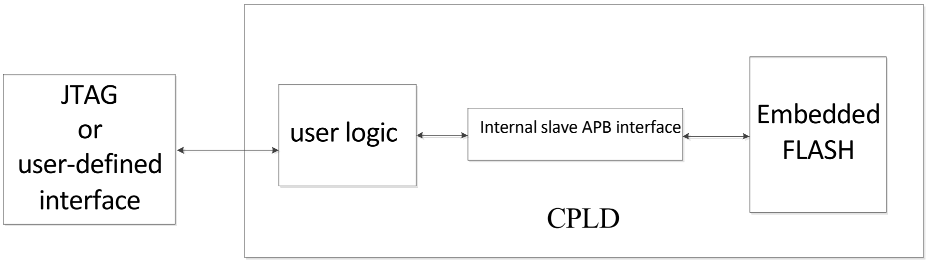


Figure 22 In-system indirect programming logic block diagram

Please refer to Appendix 7 for detailed internal operation procedures from APB.

8. Boundary scan and JTAG configuration

JTAG test refers to the application of test stimulus and analysis of test response through JTAG pins, so as to realize the fault diagnosis of the circuit under test.

The device supports IEEE1149.1 (boundary-scan testing) and IEEE1532 (programmable device in-system configuration) standards.

Boundary scan test (BST) refers to the test of digital circuits by using the boundary scan unit of the chip pins through the JTAG bus.

(1) System block diagram

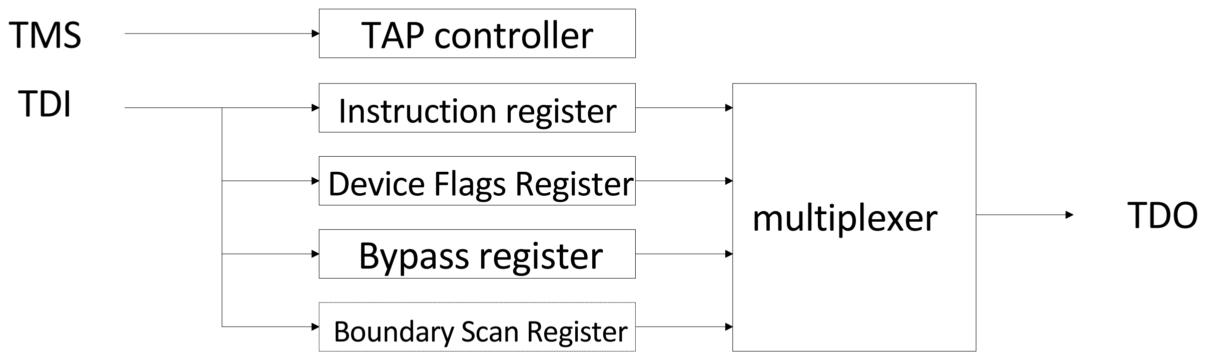


Figure 23 JTAG Boundary Scan System Block Diagram

The boundary scan hardware structure is divided into 4 parts:

Test Access Port (TAP)

Test Access Port Controller (TAPC)

Instruction Register (IR)

Test Data Register (TDR)

(2) TAP controller

The TAP controller is the core that controls the entire JTAG work. According to the TCK and TMS signal conversion, the TAP controller outputs various timings and patterns required by the instruction register and the test data register, and generates control signals for capturing, shifting, and updating the instruction register and the test data register.

The TAP controller is a 16-state synchronous finite state machine. As shown in Figure 24, the state machine is divided into 3 columns, corresponding to reset operation, test data register operation, and instruction register operation. The data on the state switch line is the TMS signal level. On the rising edge of TCK, the state of the TAP controller is toggled.

For testing, TAPC's state machine consists of 3 basic actions: stimulus (data update state), execution (test/idle state), response (data capture state). For different types of tests, all are subsets of these 3 actions. For example, the external test (EXTEST instruction), only needs two actions of excitation and response.

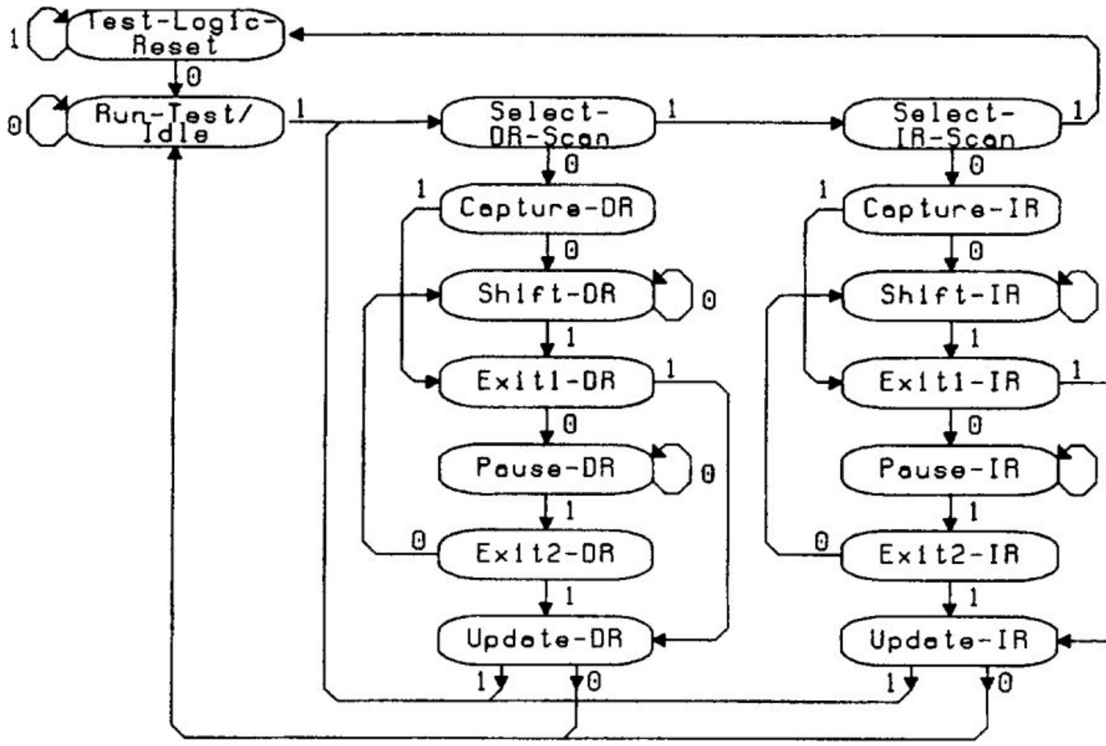


Figure 24 TAP synchronous finite state machine

Note: The detailed status introduction is shown in Appendix 8 .

(3) Instruction register

The instruction register is 10 bits long.

The instruction register receives and decodes instructions, generates control signals, selects the test data register to be placed on the scanning link from TDI to TDO, and controls the loading source and driving purpose of the test data register.

When TAPC enters the instruction capture state, at the rising edge of TCK, the instruction register captures data {1'b0, busy, wakedown_over, wakeup_over, init_n, init, isc_enabled, isc_done, 2'b01}. busy indicates that the embedded FLASH is busy, wakedown_over indicates the completion of wake-up shutdown, wakeup_over indicates the completion of wake-up, init_n indicates the completion of INIT_FLAG_N, init indicates the completion of device initialization, isc_enabled indicates that the JTAG ISC operation is enabled, and isc_done indicates that the JTAG ISC operation is completed. The 2 least significant bits are a fixed value of 01, which realizes the detection and fault location of the integrity of the scanning link.

When writing instructions to the instruction register, write the lower bits first.

When the TAPC enters the instruction update state, on the falling edge of TCK, the instruction from the shift register path is latched into the parallel output of the instruction register. Once the new command is latched, it is updated with the current command.

(4) Instruction set

Table 21 JTAG Instruction Set

INSTRUCTION	TYPE	OPCODE	DESCRIPTION
BYPASS	1149.1 non-test instruction	1111111111	Bypass instruction
SAMPLE/PRELOAD	1149.1 non-test instruction	1010000000	Sampling / Preloading Instructions
EXTEST	1149.1 test command	1010000001	external test command
INTEST	1149.1 test command	1010000010	internal test command
IDCODE	1149.1 non-test instruction	1010000011	Identification instruction
HIGHZ	1149.1 test command	1010000101	high resistance command
JRST	Design Proprietary	1010001010	reset command
READ_UID	Design Proprietary	0101001100	read UID instruction
RDSR	Design Proprietary	0101011001	Read Status Register Instruction
WADR	Design Proprietary	0101011010	write address command
ERASE	Design-only	0101011101	erase command
ERASE_PAGE	Design Proprietary	0101011110	page erase command
ERASE_CTL	Design Proprietary	0101011111	Control Erase Command
PROGRAM	Design Proprietary	0101100000	programming instructions
PROGRAM_CTL	Design Proprietary	0101100001	Control programming instructions
READ	Design Proprietary	0101100010	read command
READ_CTL	Design Proprietary	0101100011	control read command
PROGRAM_LOCK	Design Proprietary	0101100100	Lock Embedded Flash instruction
READ_LOCK	Design Proprietary	0101100101	Read embedded Flash lock flag instruction
EFlash_SLEEP	Design Proprietary	0101100110	Embedded Flash sleep command
EFlash_WAKEUP	Design Proprietary	0101100111	Embedded Flash wake up command
CFGI	Design Proprietary	1010001011	configuration directives

JWAKEUP	Design Proprietary	1010001101	wake up command
CFGO	Design Proprietary	1010001100	readback command

Note: Please refer to Appendix 1 for detailed introduction of JTAG instructions.

(5) Test Data Register (TDR)

Table 22 Test data register table

ATTRIBUTES	NAME	BIT WIDTH
Test Data Register	Device Identification Register	32
	Bypass register	1
	Boundary Scan Register	2

Table 23 Instruction and test data register correspondence table

INSTRUCTION	TEST DATA REGISTER
BYPASS	Bypass register
SAMPLE/PRELOAD	Boundary Scan Register
EXTEST	Boundary Scan Register
INTEST	Boundary Scan Register
IDCODE	Device Identification Register
USERCODE	Device Identification Register
HIGHZ	Bypass register

1. Device Identification Register

The device flags register is 32 bits long.

The device flag register can be accessed through interfaces such as JTAG, SPI and I^2C.

Table 24 Device Identification Register

BIT	NAME	DESCRIPTION
[31:28]	version	Version
[27:12]	part number	Part number
[11:1]	manufacturer identity	Manufacturer ID
[0]	1	constant 1

2. Bypass register

When some chips need to be isolated, the bypass register can be short-circuited and shorten the length of the entire boundary-scan link.

The length of the bypass register is 1 bit.

The bypass register is capable of capture and shift operations and has no parallel outputs.

When the state machine of TAPC is in the data capture state, the bypass register captures a logic 0 on the rising edge of TCK.

When the state machine of TAPC is in the data shift state, the bypass register captures TDI on the rising edge of TCK.

The bypass register has no parallel output.

3. Boundary scan registers

A boundary-scan register is a collection of boundary-scan cells. Boundary-scan cells are placed on input ports, output ports, bidirectional ports, and tri-state ports of device signals. Boundary-scan cells are connected together to form a boundary-scan register.

Pin connections and placement constraints determine the order in which boundary-scan cells are connected.

Multiplexing PAD TCK, TMS, TDI, TDO no boundary scan register.

The boundary-scan registers for all other PADs are the boundary-scan registers for bidirectional DC PADs.

Boundary scan register usage for bidirectional DC PAD: BC_2 Control+BC_7 Data

(6) Operation process

1. CRAM configuration

Configuration/reconfiguration

Configuration, the process of the CPLD chip from the beginning of configuration to enter user mode; reconfiguration, after the CPLD chip enters user mode, reset (JTAG soft reset or externally triggered INIT_FLAG_N hard reset), clear all CRAMs, exit user mode, and then the process of reconfiguration. The flow is as follows:

Load the JRST instruction into IR, write the instruction low bit first
Enter the Run-Test/Idle state for at least 1 TCK cycle
Loop into CAPTURE-IR state and load CFGI instruction into IR until init_n is detected as high, first write instruction low
Enter SHIFT-DR state, load bit stream, write high bit first
Enter Test Logic Reset state
Load the JWAKEUP instruction into IR, write the instruction low bit first
Enter Run-Test/Idle state for at least 30 TCK cycles
(wake-up by user clock for a minimum of 30 UCLK cycles)
Load CFGI instruction into IR, write instruction low bit first
Enter Run-Test/Idle state for at least 125 TCK cycles

Enter SHIFT-DR state:
Write a filler word (FFFFFFFF), write high bit first
Write synchronous word (01332D94), write high bit first
Write the packet header (00000000) with no operation, write the high bit first
Write packet header (B2400001) with read STATUSR register, write high bit first
Load the CFGO instruction into IR, write the instruction low bit first
Enter SHIFT-DR state, shift out the content of STATUSR register, high bit first out
Enter Test Logic Reset state

2. Embedded Flash programming

After the device initialization is completed, it needs to wait for 100us before operating.

Please refer to Appendix 2 for the detailed JTAG operation process. The number of continuous clock cycles in the Run Test Idle state is obtained from the fixed delay time when TCK is 50MHz. The user can determine the number of clocks according to the actual TCK frequency. Users can determine the number of clocks to wait according to the actual TCK frequency. If TCK needs 100 cycles when it is 50MHz, then it needs 10 clock cycles when it is 5MHz.

3. Step test

The single-step test controls the input port through the boundary scan register, applies excitation to the input port of the chip, and captures the response data of the chip output port into the boundary scan register, and serially shifts it out through the TDO port.

Table 25 Step test

Load preload command (PRELOAD)
Send first test vector (input stimulus serially shifted in and parallel updated to boundary scan registers)
Load In-Test Instruction (INTEST) (input changed from input port to boundary scan register)
While sending the second test vector, move out the first response data from the TDO port for response analysis and fault diagnosis (the input stimulus is serially shifted in while the output response is serially shifted out, and then the input stimulus is updated to the boundary scan register in parallel)
While sending the third test vector, move out the second response data from the TDO port for response analysis and fault diagnosis (the input stimulus is serially shifted in while the output response is serially shifted out, and then the input stimulus is updated to the boundary scan register in parallel)
...
While sending the n+1th test vector, move out the nth response data from the TDO port for response analysis and fault diagnosis (the input stimulus is serially shifted in while the output response is serially shifted out, and then the input stimulus is updated to the boundary scan register in parallel)

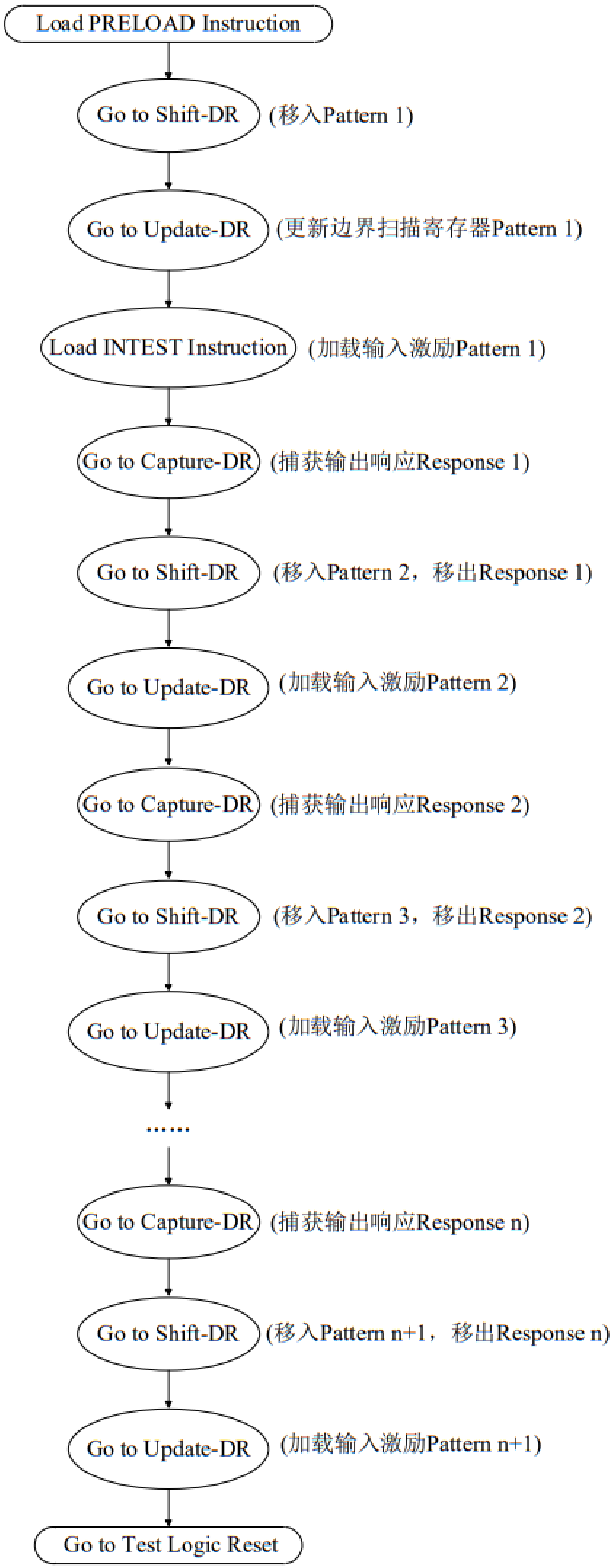


Figure 25 Boundary Scan Single Step Test

4. Interconnect Test

The interconnection test controls the output port through the boundary scan register, and generates the output stimulus of the output port of the chip. The response data from the input port of the chip is captured into the boundary scan register and shifted out serially by the TDO port.

Table 26 Interconnect test

Load preload command (PRELOAD)
Send first test vector (output stimuli serially shifted in and parallel updated to boundary scan registers)
Load external test instruction (EXTEST) (output controlled from output port to boundary scan register)
While sending the second test vector, move out the first response data from the TDO port for response analysis and fault diagnosis (the output stimulus is serially shifted in while the input response is serially shifted out, and then the output stimulus is updated to the boundary scan register in parallel)
While sending the third test vector, move out the second response data from the TDO port for response analysis and fault diagnosis (the output stimulus is serially shifted in while the input response is serially shifted out, and then the output stimulus is updated to the boundary scan register in parallel)
...
While sending the n+1th test vector, move out the nth response data from the TDO port for response analysis and fault diagnosis (the output stimulus is serially shifted in while the input response is serially shifted out, and then the output stimulus is updated to the boundary scan register in parallel)

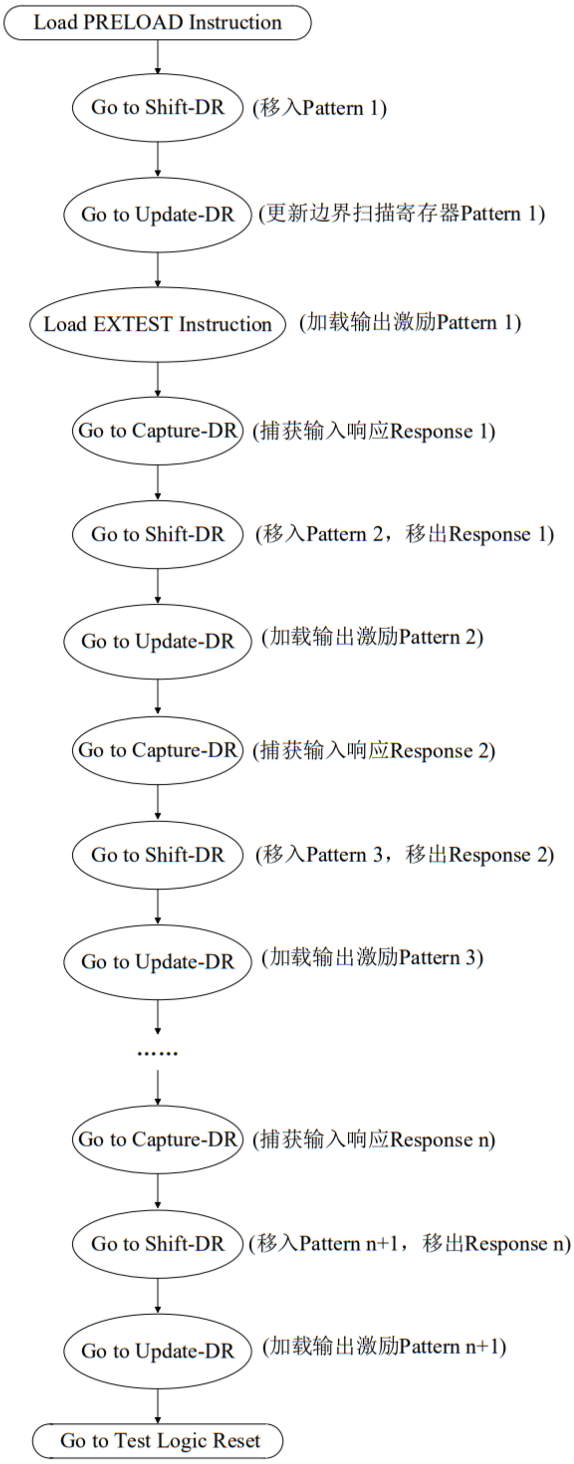


Figure 26 Boundary Scan Interconnect Test

9. User logic interface

The CPLD device supports 6 user logic interfaces, which are JTAG user command interface, UID interface, power controller, user wake-up interface, internal slave APB interface, and readback CRC. For detailed GTP usage, please refer to the PDS software installation appendix .. .\PDS_xxx\arch\vendor\pango\verilog\simulation.

(1) JTAG user command interface (SCANCHAIN)

TCK, TMS, TDI are input from PAD to Fabric.

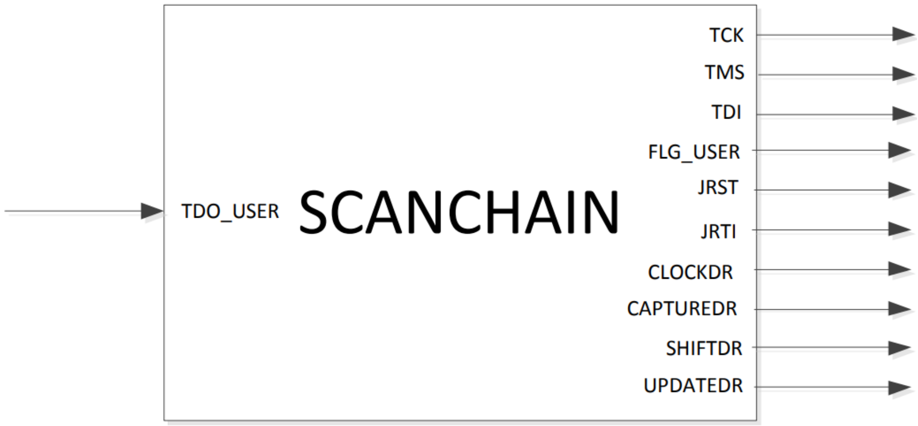


Figure 27 JTAG user command interface

1. Port list

Table 27 JTAG user port list

NAME	I/O	DESCRIPTION
TCK	O	JTAG Clock, up to 50MHz
TMS	O	JTAG Test mode selection. Clock TCK Rising edge, control TAP Controller state jump
TDI	O	JTAG test data input. Serial input to JTAG instruction and data registers on rising edge of clock TCK
FLG_USER	O	User Defined Register Instruction Flags
TDO_USER	I	User-defined register data input
JRST	O	User JTAG soft reset
JRTI	O	User JTAG test / idle
CLOCKDR	O	Gated clock for user-defined register instruction operations
CAPTUREDR	O	User CAPTURE-DR flag
SHIFTDR	O	User SHIFT-DR flag
UPDATEDR	O	User UPDATE-DR ID

2. Interface timing

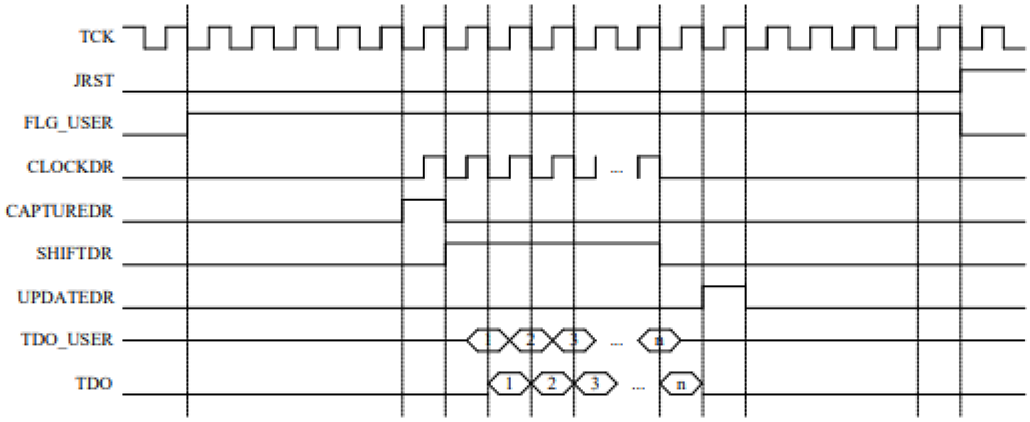


Figure 28 JTAG User Interface Timing Diagram

(2) UID interface (UDID)

The 64-bit UID is stored in the embedded Flash and programmed uniformly when the device leaves the factory. Every time the device is powered on, the UID in the embedded Flash is automatically read into the register so that the user can take it out at any time.

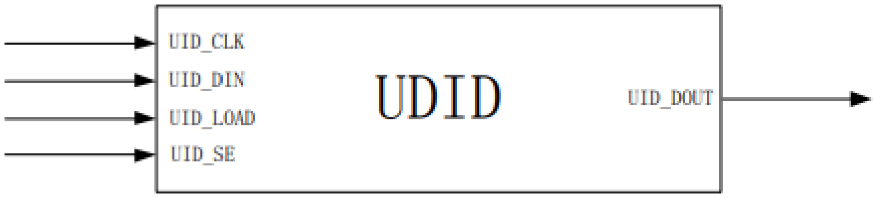


Figure 29 UID interface diagram

1. Port list

Table 28 UID port list

NAME	I/O	DESCRIPTION
UID_CLK	I	UID clock
UID_DIN	I	UID serial data input
UID_LOAD	I	UID Parallel Data Loading
UID_SE	I	UID serial data shift
UID_DOUT	O	UID serial data out

2. Interface timing

UID length is 64 bit.

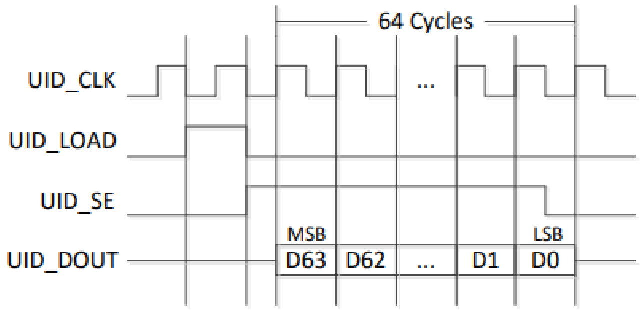


Figure 30 64 -bit UID interface timing diagram

User can extend UID bit width.

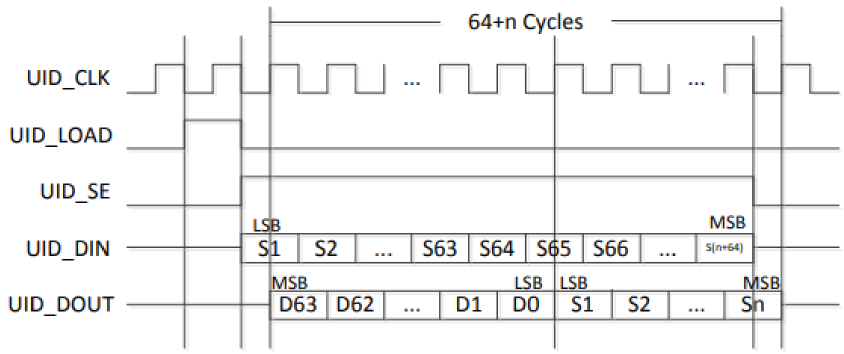


Figure 31 Extended UID Timing diagram

(3) Power consumption controller

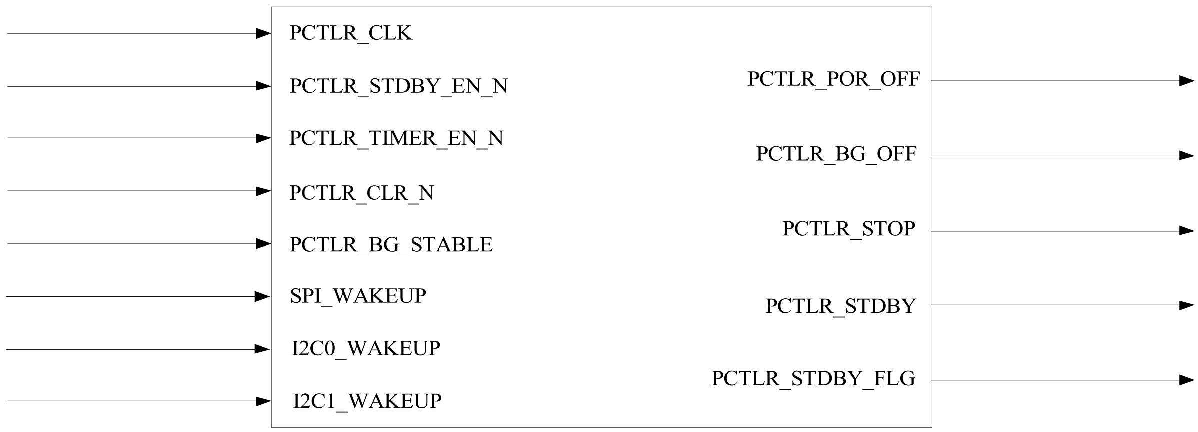


Figure 32 Power Consumption Controller Interface Diagram

1. Port list

Table 29 Power Controller Port List

NAME	I/O	SOURCE / DESTINATION	DESCRIPTION
PCTLR_CLK	I	SRB	Clock, the highest frequency is 50M
PCTLR_STDBY_EN_N	I	SRB	Standby enable, active low starts from the falling edge of standby enable, and enters the shutdown process from working mode to standby mode. Starting from the rising edge of standby enable, it enters the wake-up process from standby mode to working mode.
PCTLR_TIMER_EN_N	I	SRB	Timer enable, low active pulse signal, start timer to start counting
PCTLR_CLR_N	I	SRB	End of standby, low effective pulse signal, used to clear the standby flag after wake-up
PCTLR_BG_STABLE	I	bandgap circuit	bandgap circuit stability
PCTLR_POR_OFF	O	Power-on reset circuit	Power-on reset circuit shuts down
PCTLR_BG_OFF	O	bandgap circuit	When the bandgap circuit is turned off, the power-on reset circuit and analog circuits (PLL, OSC, differential IO) are turned off when the bandgap circuit is turned off
PCTLR_STOP	O	SRB	stop Used for standby preparation. User logic prepares for standby by turning off signals such as clocks
PCTLR_STDBY	O	SRB	standby Global signals, via SRB Output to user logic, IO PAD , PLL . Used to control the logic circuit into standby mode
PCTLR_STDBY_FLG	O	SRB	standby sign Indicates that the device is in standby mode
SPI_WAKEUP	I	SPI function block	SPI function block wakeup request When the wake-up enable bit of the control register of the SPI function block is valid, if the SPI function block is selected as a slave device (the chip select input is valid), then the SPI function block sends a wake-up request to the power controller
I2C0_WAKEUP	I	I^2C0 function block	I^2C0 function block wakeup request When the wake-up enable bit of the control register of the I^2C0 function block is valid, if the I^2C0 function block is selected as a slave device (the slave device address matches), then the I^2C0 function block sends a wake-up request to the power controller

I2C1_WAKEUP	I	I^2C1 function block	I^2C1 Function Block Wakeup Request When the wake-up enable bit of the control register of the I^2C1 function block is valid, if the I^2C1 function block is selected as a slave device (the slave device address matches), then the I^2C1 function block sends a wake-up request to the power controller

2. Mode conversion

The power controller has four modes, which are shutdown mode, standby mode, wake-up mode, and working mode. The state machine composed of PCTLR_STOP and PCTLR_STDBY signals is used for mode switching control. The mode transitions and truth table are shown below.

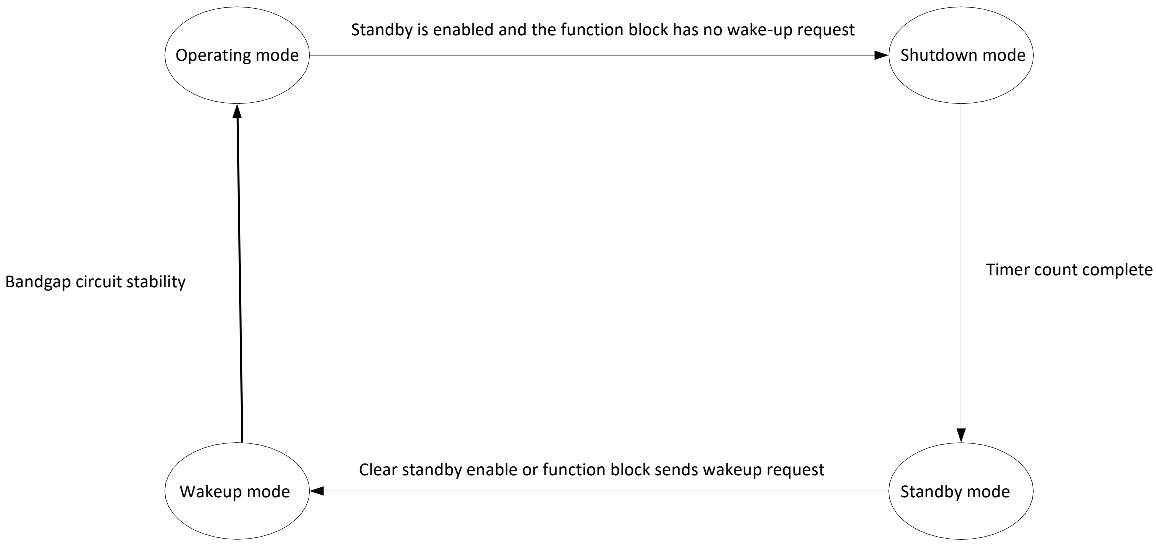


Figure 33 Power Controller Mode Transition Diagram

Table 30 Power Controller Mode Truth Table

	PCTLR_STOP	PCTLR_STDBY
Operating mode	0	0
Shutdown mode	1	0
Standby mode	1	1
Wake up mode	0	1

3. Interface timing

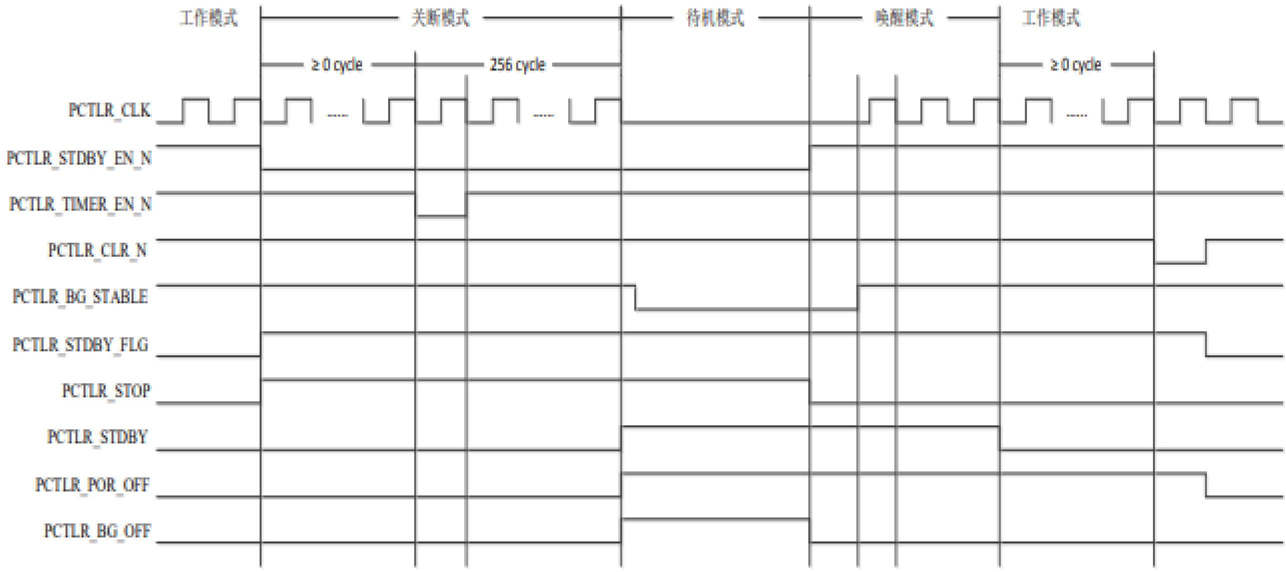


Figure 34 Timing Diagram of Power Consumption Controller Interface

Note that only after the bandgap circuit is stable can there be a stable clock.

1) standby

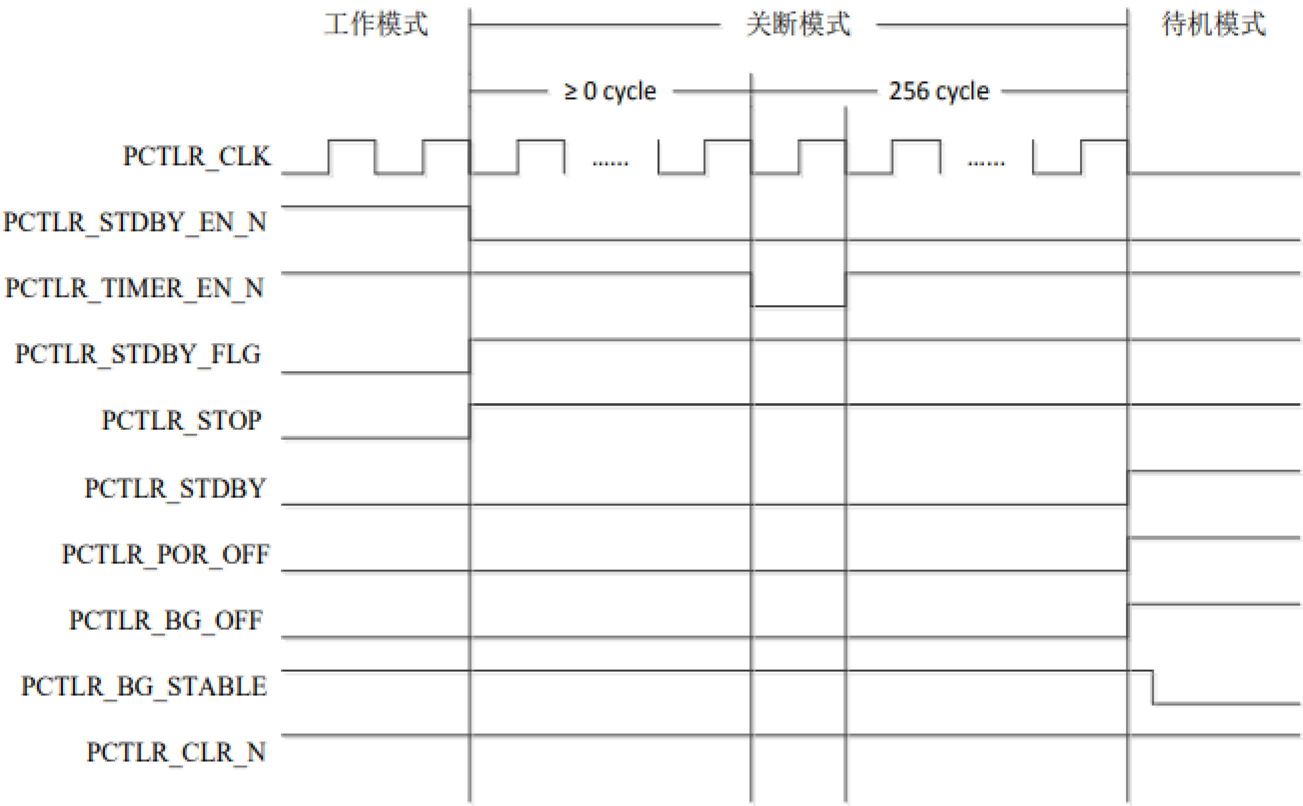


Figure 35 Standby Sequence Diagram

1. Standby enable STDBY_EN_N changes from invalid to valid, and the power controller enters shutdown mode from working mode:

a. The standby flag, STDBY_FLG, is generated to indicate that the device is in standby mode.

b. A standby preparation STOP is generated, and the user logic prepares for standby by turning off signals such as clocks.

2. When the timer is enabled, the TIMER_EN_N pulse is valid, and the timer starts counting.

3. After the timer counts 256 clock cycles:

a. Generate a standby global signal, output to the user logic, IO PAD, OSC, PLL through SRB, and use it to control the logic circuit to enter the standby mode.

b. Turn off the bandgap circuit and power-on reset circuit. When the bandgap circuit is turned off, the power-on reset circuit and analog circuits (PLL, OSC, differential IO) are turned off.

2) wake

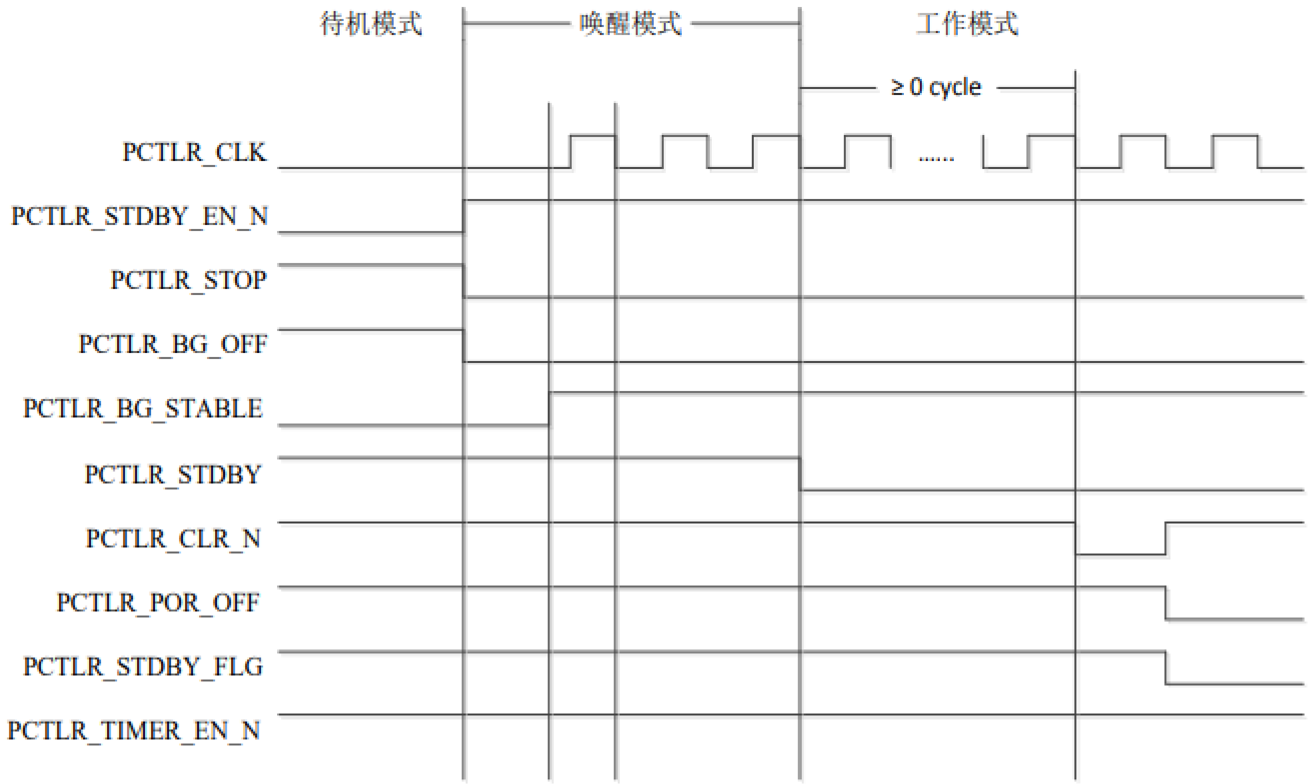


Figure 36 Wake-up Timing Diagram

1. Standby enable STDBY_EN_N changes from valid to invalid, and the power controller enters wake-up mode from standby mode:

a. Clear STOP for standby preparation.

b. Turn on the bandgap circuit.

2. After the bandgap circuit is stable, the standby global signal is cleared, and the power controller enters the working mode from the wake-up mode.

3. After entering the working mode, the user logic generates a standby end pulse:

a. Turn on the power-on reset circuit.

b. Clear the standby flag STDBY_FLG.

3) SPI / I^{2}C function block wakes up

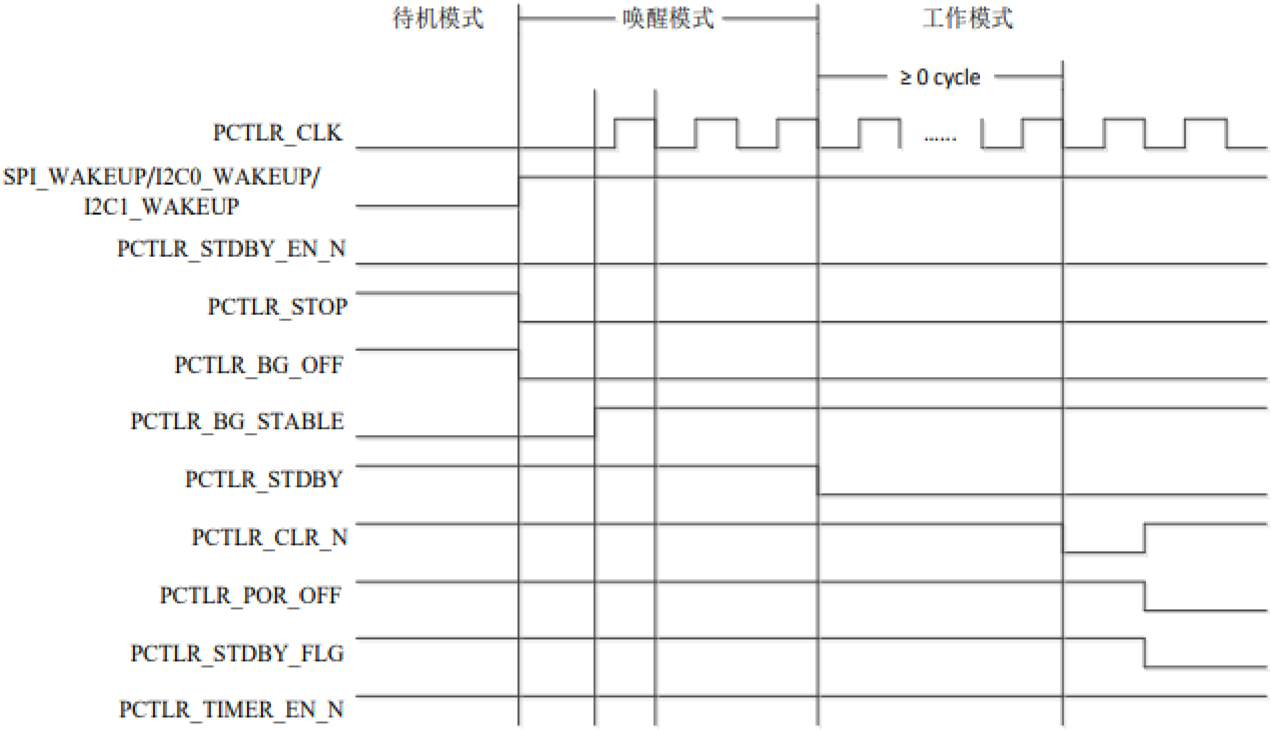


Figure 37 SPI/I^{2}C function block wake-up timing diagram

When the wake-up enable bit of the control register of the SPI function block is valid, if the SPI function block is selected as a slave device (the chip select input is valid), then the SPI function block sends a wake-up request to the power controller.

When the wake-up enable bit of the control register of the I^{2}C function block is valid, if the I^{2}C function block is selected as a slave device (the slave device address matches), then the I^{2}C function block sends a wake-up request to the power controller

1. The function block sends a wakeup request (SPI_WAKEUP or I2C0_WAKEUP or I2C1_WAKEUP), and the power controller enters wakeup mode from standby mode:

a. Clear STOP for standby preparation.

b. Turn on the bandgap circuit.

2. After the bandgap circuit is stable, the standby global signal is cleared, and the power controller enters the working mode from the wake-up mode.

3. After entering the working mode, the user logic generates a standby end pulse:

a. Turn on the power-on reset circuit.

b. Clear the standby flag STDBY_FLG.

4) SPI/I^{2}C function block standby

After the SPI/I^{2}C function block operates in slave mode, cancel the wake-up request. At this time, if the user standby enable is valid and there is no wake-up request from other functional blocks, the power consumption controller returns to the standby mode.

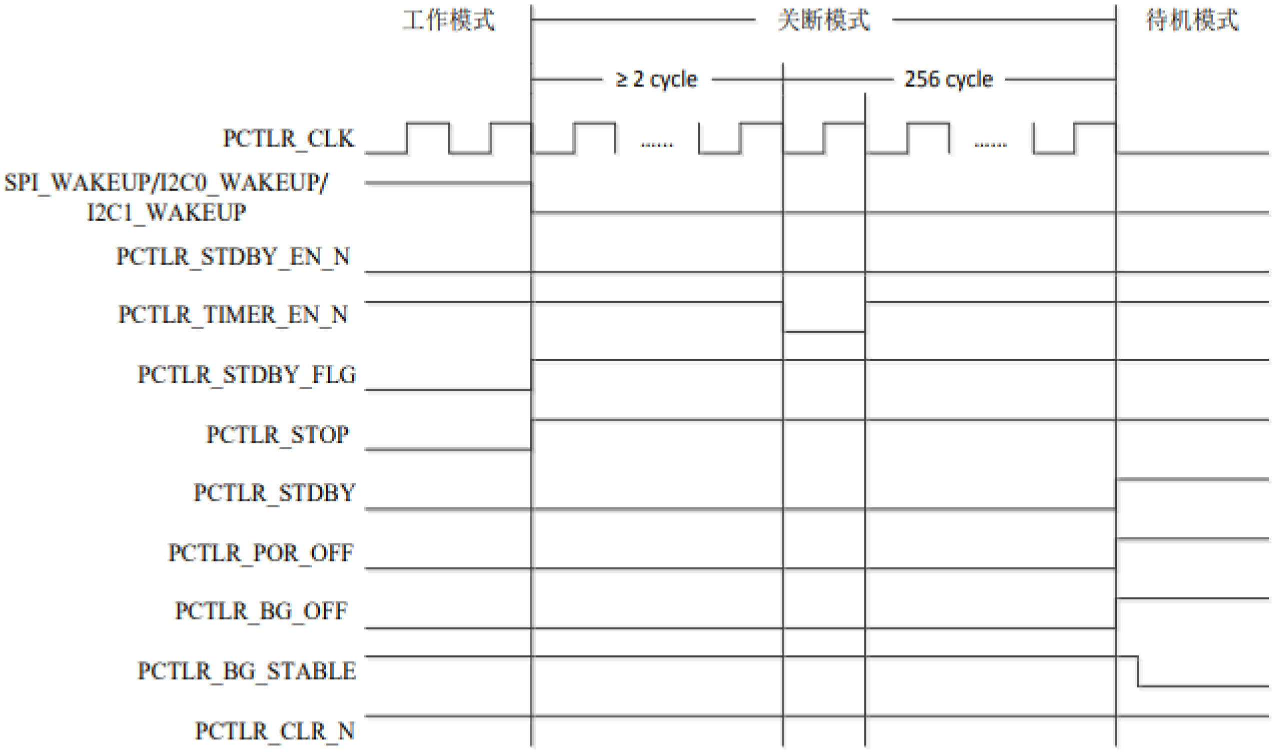


Figure 38 SPI/I^{2}C function block standby timing diagram

1. Standby enable STDBY_EN_N changes from invalid to valid, and the power controller enters shutdown mode from working mode:

a. The standby flag STDBY_FLG is generated to indicate that the device is in standby mode.

b. Generate STOP for standby preparation, and the user logic prepares for standby by turning off the clock and other signals.

2. When the timer is enabled, the TIMER_EN_N pulse is valid, and the timer starts counting.

3. After the timer counts 256 clock cycles:

a. Generate standby global signal, output to user logic, IO PAD, OSC, PLL through SRB, used to control logic circuit to enter standby mode.

b. Turn off the bandgap circuit and power-on reset circuit. When the bandgap circuit is turned off, the power-on reset circuit and analog circuits (PLL, OSC, differential IO) are turned off.

(4) User wake-up interface (START)



Figure 39 User wake-up interface diagram

1. Port list

Table 31 User wakeup port list

NAME	I/O	DESCRIPTION
UCLK	I	User clock
GOUTEN_I	I	User output enable
GRSN_I	I	User Global Reset
GWEN_I	I	User memory write enable

2. Parameter

Table 32 User wakeup parameter table

NAME	DEFAULTS	DESCRIPTION
CP_GOUTEN_EN	DISABLE	User output enable is valid ENABLE: valid DISABLE: disabled
CP_GRSN_EN	DISABLE	User global reset is valid ENABLE: valid DISABLE: disabled
CP_GWEN_EN	DISABLE	User memory write enable is valid ENABLE: valid DISABLE: disabled

(5) Internal slave APB interface

The internal slave APB interface of the configuration control system is an APB slave device interface, which is compatible with the AMBA APB3 protocol and supports 7 slave devices, namely CCS, SPI, I^2C0, I^2C1, TIMER, PLL0, and PLL1.

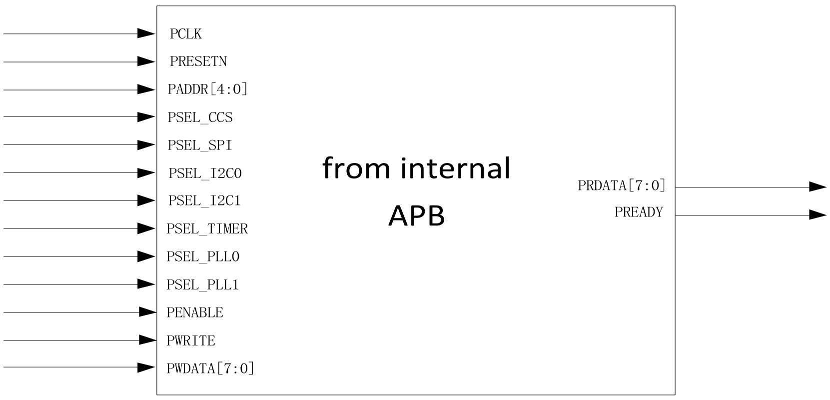


Figure 40 From the internal APB interface diagram

1. Port list

Table 33 From internal APB port list

NAME	I/O	DESCRIPTION
pclk	I	clock, rising edge sampling
preset	I	Asynchronous reset, active low
paddr[4:0]	I	address bus
psel	I	Indicates that at the current moment, only one of the seven slave devices can be selected
penable	I	Enable Indicates the second and subsequent cycles of the transfer.
pwrite	I	Direction 0 : read 1 : write
pwdata[7:0]	I	data bus input
prdata[7:0]	o	data bus output
ready	o	get ready Indicates the end of a normal bus cycle

2. Interface timing

Write

no wait write

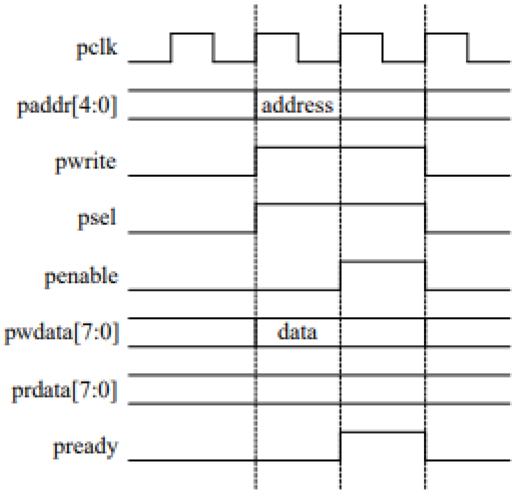


Figure 41 no wait write

With wait cycle

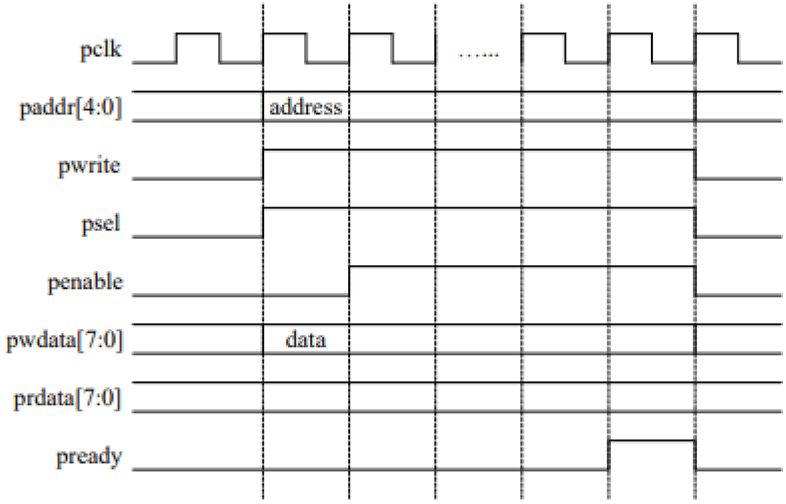


Figure 42 Waiting to write

Continuous writing

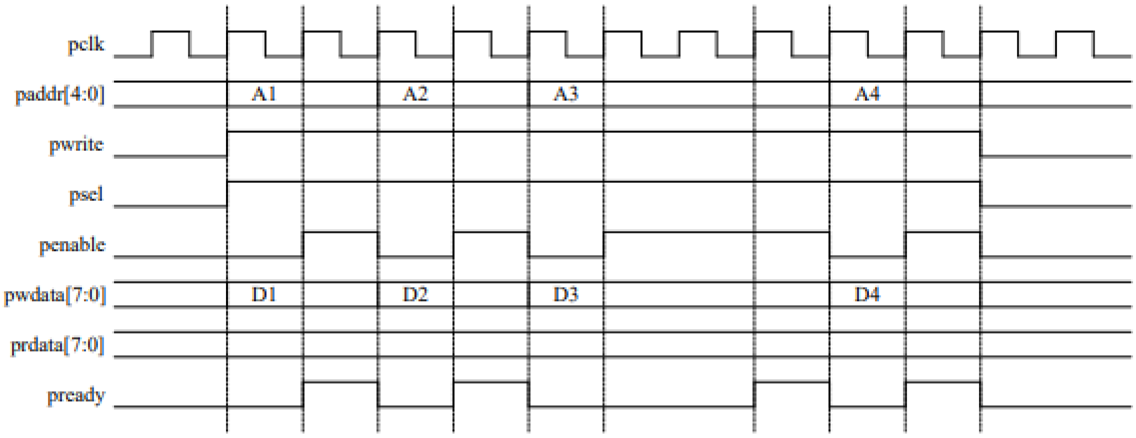


Figure 43 Continuous writing

Read

no wait read

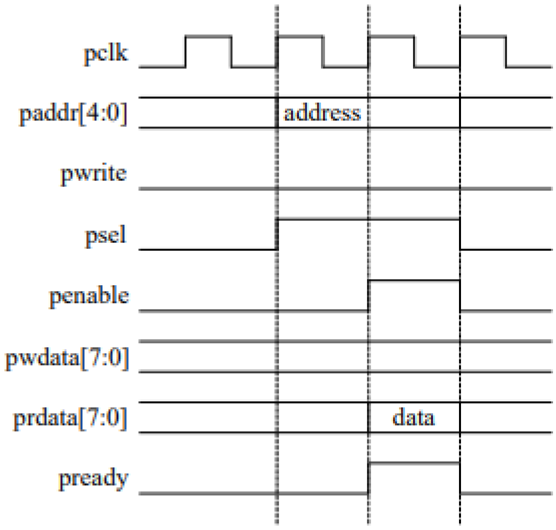


Figure 44 Read without wait

With wait to read

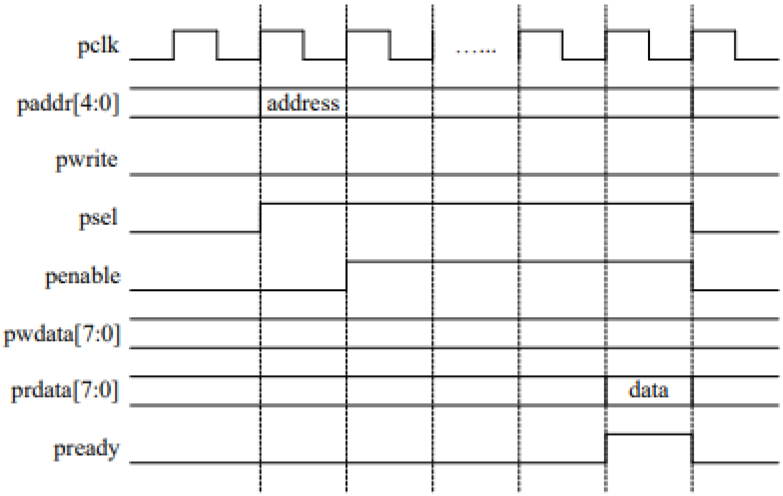


Figure 45 Waiting to read

Continuous reading

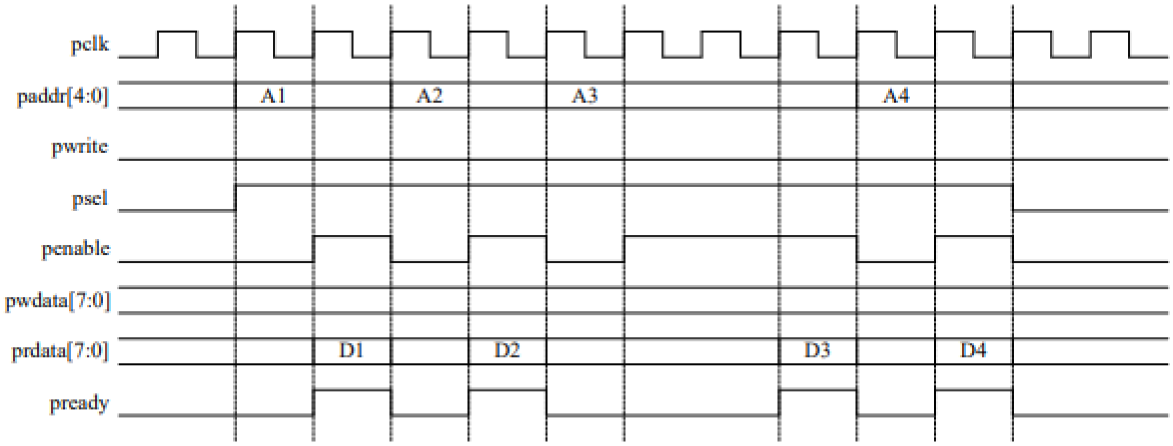


Figure 46 Continuous reading

Back to back

read after write

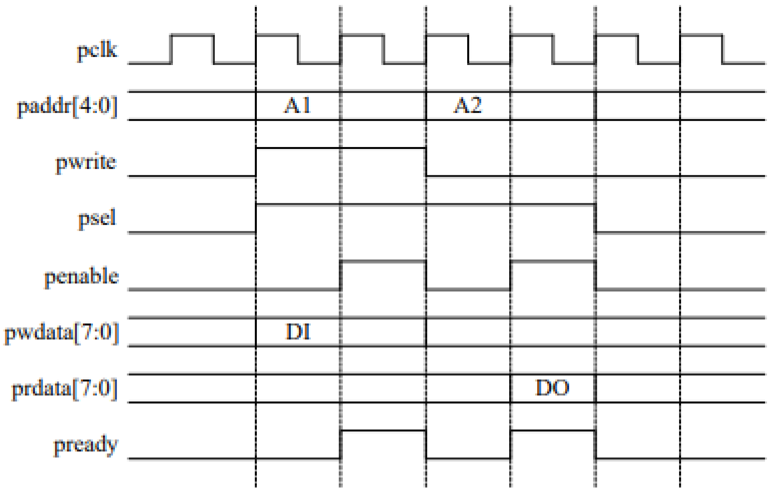


Figure 47 Read after write

read after write

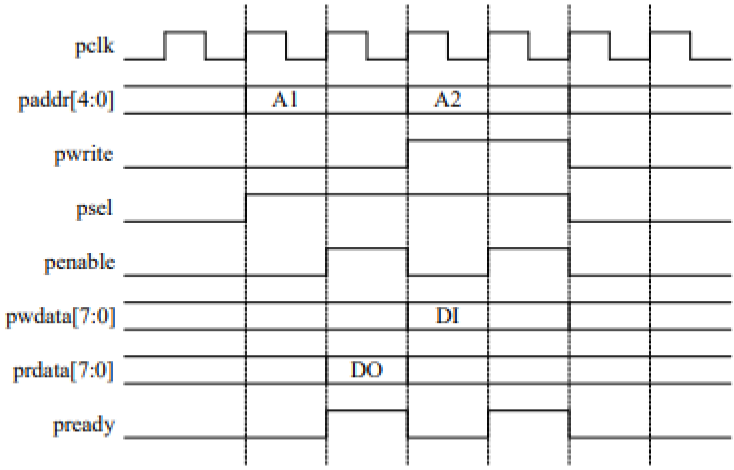


Figure 48 Read after write

3. Register

Table 34 Internal slave APB register

NAME	R/W	ADDRESS	DESCRIPTION
IDCODER0	R	00000	IDCODE register 0
IDCODER1	R	00001	IDCODE register 1
IDCODER2	R	00010	IDCODE register 2
IDCODER3	R	00011	IDCODE Register 3
USERCODER0	R	00100	USERCODE register 0
USERCODER1	R	00101	USERCODE register 1
USERCODER2	R	00110	USERCODE register 2
USERCODER3	R	00111	USERCODE Register 3
UIDR0	R	01000	UID register 0
UIDR1	R	01001	UID register 1
UIDR2	R	01010	UID register 2
UIDR3	R	01011	UID Register 3
UIDR4	R	01100	UID Register 4
UIDR5	R	01101	UID Register 5
UIDR6	R	01110	UID Register 6
UIDR7	R	01111	UID Register 7
STATUSR0	R	10000	Configuration Control System Status Register 0
STATUSR1	R	10001	Configuration Control System Status Register 1
STATUSR2	R	10010	Configuration Control System Status Register 2
STATUSR3	R	10011	Configuration Control System Status Register 3
IRQCTLR	R/W	10100	Interrupt Control Register
CMDR	R/W	10101	command register
ADR0	R/W	10110	address register 0
ADR1	R/W	10111	address register 1
ADR2	R/W	11000	address register 2
DATATR	W	11001	data send register

DATARR	R	11010	Data Receive Register
STATUSR	R	11011	status register
IRQSTATUSR	R	11100	Interrupt Status Register
IRQR	R	11101	interrupt register
LOCKRAR0	R	11110	Lock Page Register 0
LOCKRAR1	R	11111	Lock Page Register 1

Note: For the introduction of internal slave APB interface registers, please refer to Appendix 6 .

(6) Readback CRC (RBCRC)

The readback CRC check uses a 32-bit CRC algorithm, and the initial value of the CRC is 0. When reading back, do a CRC calculation for 32-bit data.

A CRC value is calculated for each readback. If the CRC result of the readback calculation is inconsistent with the value of the readback CRC register, the readback CRC error bit of the status register is high.

The readback CRC clock is input from OSC to CCS and SRB simultaneously.

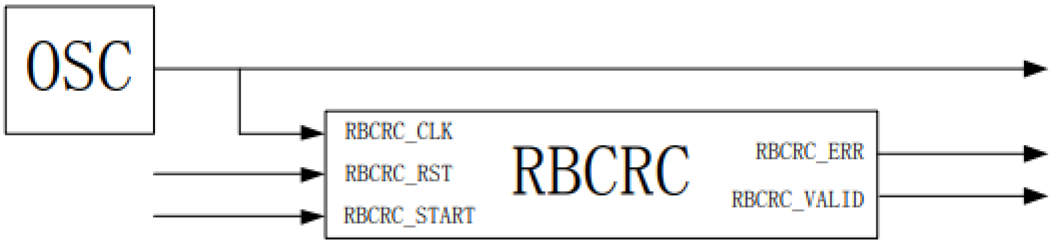


Figure 49 Readback CRC interface diagram

1. Port list

Table 35 Read back CRC port list

NAME	I/O	SOURCE / DESTINATION	DESCRIPTION
RBCRC_CLK	I	OSC	Readback CRC Clock, the default is 2.08M , the highest frequency is 33.3M
RBCRC_RST	I	SRB	Readback CRC Reset Used to reset the readback CRC error flag and clear the readback CRC initial value When readback CRC reset and readback CRC start are valid at the same time, readback CRC start cannot be started Readback CRC reset is invalid, when RBCRC_START is valid, start readback CRC start The reset and read-back CRC operation can only be performed before and after the read-back CRC is completed. During the read-back CRC process (from the start of the read-back CRC to the read-back CRC information valid flag indicating the end of the read-back CRC), the reset and read-back operation cannot be performed. Read CRC operation

RBCRC_START	I	SRB	Readback CRC start Pulse Start Readback CRC Start During the process of reading back CRC (from the start of reading back CRC to the end of reading back CRC indicated by the valid flag of reading back CRC information), the start signal of reading back CRC is invalid. The next read-back CRC can only be started after the end of one read-back CRC
RBCRC_ERR	O	SRB	Readback CRC error flag
RBCRC_VALID	O	SRB	Readback CRC information valid sign Pulse signal, indicating readback CRC Error flag valid and readback CRC the end

2. Interface timing

1) single step operation

After the last CRC readback ends, before the next CRC readback starts.

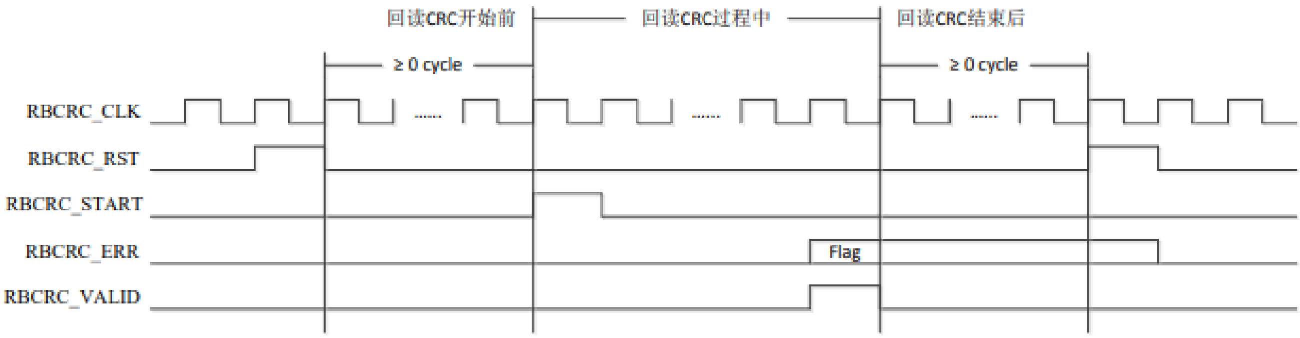
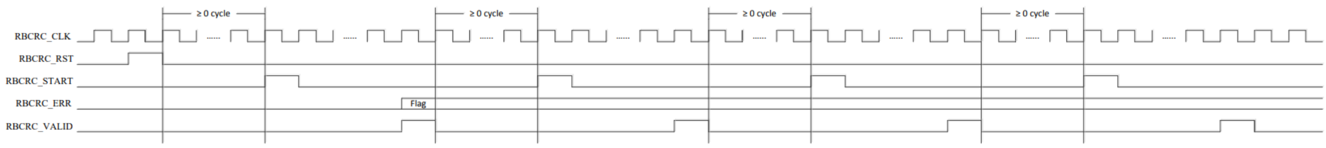


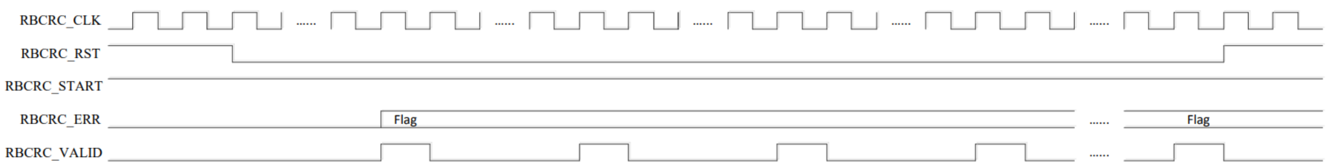
Figure 50 Timing diagram of single-step operation

2) Continuous operation



3) Burst operation

When reading back CRC continuously, the RBCRC_START signal can be fixed to 1.



10. Embedded hard core

CPLD devices support SPI, I2C and timer functions. For details, please refer to "UG030007_Compact Series CPLD Embedded Hard Core User Guide"

11 . Contact us

Контакты для технических и коммерческих вопросов

ООО «Инэк»

г. Санкт-Петербург, ул. Яблочкова, д. 20, литер А, оф. 504

contact@innek.ru

+7 (812) 200-40-37

Комментарии и предложения по переводу пишите на почту

Appendix 1

Introduction to JTAG interface instructions. When writing an instruction, write the low bit first.

BYPASS

Bypass instructions.

The bypass instruction puts the bypass register on the scan chain of TDI and TDO, and the device is in active mode.

SAMPLE/PRELOAD

Sample/preload instructions.

The sample/preload instruction places the boundary-scan registers on the TDI-TDO scan link.

When the state machine of TAPC is in the data capture state, it captures the data of the input pins of the device.

When the state machine of TAPC is in the data shift state, the test vector is moved into the boundary scan register through the scan chain.

When the state machine of TAPC is in the state of data update, the newly moved data will be latched to the parallel output terminal of the boundary scan unit, but will not be updated to the output pin of the device.

In Sample/Preload mode, the device is in Active mode for normal operation.

EXTEST

external test command.

The external test command realizes the interconnection test between the chip pins.

The external test instruction places the boundary-scan register on the TDI-TDO scan link.

When the state machine of TAPC is in the data capture state, it captures the data of the input pins of the device.

When the state machine of TAPC is in the data shift state, the test vector is moved into the boundary scan register while the data capture result of the input pin of the device is shifted out through the scan chain.

When the state machine of TAPC is in the state of data update, it latches the newly moved data into the parallel output terminal of the boundary scan unit and updates it to the output pin of the device.

Before performing the external test command, it is necessary to perform pre-load command operation and pre-load test vectors. Once the external test command is activated, the pre-installed test vector will be driven to the output pin, so as to ensure the state of the output pin is controlled.

In the external test command mode, the device is in the test mode and cannot work normally.

INTEST

Internal test instructions.

The internal test instruction uses the boundary-scan register as the input and output of the device to implement a static test of the system logic.

The inner test instruction places the boundary-scan registers on the TDI-TDO scan link.

When the state machine of TAPC is in the data capture state, it captures the data of the output pins of the device.

When the state machine of TAPC is in the state of data shifting, the data capture result of the output pin of the device is shifted out through the scan chain, and at the same time, the test vector is shifted into the boundary scan register.

When the state machine of TAPC is in the state of data update, the newly moved data is latched to the parallel output terminal of the boundary scan unit and updated to the input pin of the device.

Before performing the internal test command, it is necessary to perform pre-load command operation and pre-load test vectors. Once the built-in test command is active, the preloaded test vectors are driven to the input pins to ensure that the state of the input pins is under control.

Clock signals are generated in the same way as non-clock signals, shifted in through the boundary-scan path.

IDCODE

Identify directives.

The identification command is used to read the device identification code.

The identification instruction places the device identification register on the TDI-TDO scan link.

When the state machine of TAPC is in the data capture state, the device identification register captures the device identification code on the rising edge of TCK.

When the state machine of TAPC is in the data shift state, on the rising edge of TCK, the device identification register is shifted one bit to the right.

HIGHZ

High resistance command.

The high-impedance command sets the device tri-state pin output to a high-impedance state.

A high-impedance instruction places a bypass register on the TDI-TDO scan link.

JRST

Reset command.

The reset instruction resets the system outside of the JTAG circuitry.

A user command places a bypass register on the TDI-TDO scan link.

When the TAPC's state machine is in the test/idle state, a reset operation is performed.

READ_UID

Read UID command.

The read UID instruction is used to read UID, the lower bit is first out.

The Read UID command places the UID register on the TDI-TDO scan link.

When the state machine of TAPC is in the data capture state, at the rising edge of TCK, the UID register captures the UID stored in the eFuse.

When the state machine of TAPC is in the data shifting state, the UID register is shifted one bit to the right

on the rising edge of TCK.

RDSR

Read status register instruction.

The read status register command is used to read the status register, the low bit is first out.

The read status register command places the status register on the TDI-TDO scan link.

When the state machine of TAPC is in the data capture state, at the rising edge of TCK, the state register captures the value of the state register that configures the control system.

When the state machine of TAPC is in the data shift state, the state register is shifted one bit to the right on the rising edge of TCK.

WADR

Write address command.

Write address command is used to write the embedded Flash start address.

The write address command places the address register on the TDI-TDO scan link.

When the state machine of TAPC is in the data capture state, on the rising edge of TCK, the address register captures the start address of the current embedded Flash.

When the state machine of TAPC is in the data shifting state, the address register is shifted one bit to the right on the rising edge of TCK.

When the state machine of TAPC is in the data update state, at the falling edge of TCK, the data from the shift register path is latched to the parallel output of the address register.

ERASE

Erase command.

Erase command erases all common pages of embedded Flash and clears the embedded Flash lock flag.

The erase command places the bypass register on the TDI-TDO scan chain.

The erase operation is performed while the state machine of the TAPC is in the test/idle state.

ERASE_PAGE

page erase command.

The page erase command erases the normal page of the embedded Flash.

The page erase instruction places a bypass register on the TDI-TDO scan link.

The page erase operation is performed while the state machine of the TAPC is in the test/idle state.

ERASE_CTL

Controls the erase command.

The control erase command erases the feature control page of the embedded Flash.

The Control Erase command places a bypass register on the TDI-TDO scan chain.

The state machine of the TAPC controls the execution of erase operations when it is in the test/idle state.

PROGRAM

programming instructions.

The programming command programs the ordinary page of the embedded Flash.

The programming instruction places a bypass register on the TDI-TDO scan link.

When the state machine of TAPC is in the data shifting state, a page of data is serially shifted into the bypass register on the rising edge of TCK.

PROGRAM_CTL

Control programming instructions.

Control programming instructions program the feature control bits of the embedded Flash.

Control programming instructions place feature control registers on the TDI-TDO scan link.

When the state machine of TAPC is in the data capture state, at the rising edge of TCK, the feature control register captures the feature control bits stored in the embedded Flash.

When the state machine of TAPC is in the data shift state, on the rising edge of TCK, the feature control register is shifted one bit to the right.

When the state machine of TAPC is in the data update state, at the falling edge of TCK, the data from the shift register path is latched to the parallel output of the feature control register.

READ

read command.

The read command is used to read the content of the embedded Flash, the low bit is first out.

The read command selects the programming register, but does not place the programming register on the TDI-TDO scan link.

When the state machine of TAPC is in the data shifting state, at the rising edge of TCK, the general content of the embedded Flash is serially shifted out from TDO through the programming register.

READ_CTL

Control read command.

The control read command is used to read the feature control bits of the embedded Flash, and the lower bits are first out.

Control programming instructions place feature control registers on the TDI-TDO scan link.

When the state machine of TAPC is in the data capture state, at the rising edge of TCK, the feature control register captures the feature control bits stored in the embedded Flash.

When the state machine of TAPC is in the data shift state, on the rising edge of TCK, the feature control register is shifted one bit to the right.

PROGRAM_LOCK

Lock instruction.

The lock command is used to program the lock flag and the last page address of the bit stream into the embedded Flash, and prohibit the programming and reading operations of the bit stream.

The lock instruction places the lock register on the TDI-TDO scan link.

When the state machine of TAPC is in the data capture state, on the rising edge of TCK, the lock register captures the lock flag and the last page address of the bit stream.

When the state machine of TAPC is in the data shift state, the lock register is shifted one bit to the right on the rising edge of TCK.

When the state machine of TAPC is in the test/idle state, it executes the programming lock flag operation, and programs the lock flag and the last page address of the bit stream into the embedded Flash.

READ_LOCK

Read lock flag instruction.

The read lock flag instruction is used to read the lock flag and the last page address of the bitstream.

The Read Lock Flags instruction places the lock register on the TDI-TDO scan link.

When the state machine of TAPC is in the data capture state, on the rising edge of TCK, the lock register captures the lock flag and the last page address of the bit stream.

When the state machine of TAPC is in the data shift state, the lock register is shifted one bit to the right on the rising edge of TCK.

EFlash_SLEEP

Embedded Flash sleep command.

The embedded Flash dormancy command sleeps the embedded Flash.

The embedded Flash dormancy command places the bypass register on the scan link of TDI-TDO.

When the TAPC's state machine is in the test/idle state, the hibernate embedded Flash operation executes.

EFlash_WAKEUP

Embedded Flash wake-up command.

The embedded Flash wake-up command wakes up the embedded Flash.

The embedded Flash wake-up command places the bypass register on the scan link of TDI-TDO.

When the state machine of TAPC is in the test/idle state, it wakes up the embedded Flash operation execution.

CFGI

Configuration directives.

Configuration instructions are used to configure the contents of the configuration memory and configuration registers.

The configure instruction places a bypass register on the TDI-TDO scan link.

When the state machine of TAPC is in the data shift state, the bit stream is serially shifted into the bypass register on the rising edge of TCK.

CFGO

Readback command.

The readback instruction is used to read back the contents of the configuration memory and configuration registers, low-order first-out.

The configuration register is selected by the readback command, and the configuration register is not placed on the scan link of TDI-TDO.

When the state machine of TAPC is in the data shift state, at the rising edge of TCK, the contents of the configuration memory and the configuration register pass through the configuration register and are serially shifted out from TDO.

JWAKEUP

wakeup command.

The wakeup command is used to wake up the device.

The wake-up instruction places a bypass register on the TDI-TDO scan link.

When the state machine of the TAPC is in the test/idle state, the wakeup operation is performed.

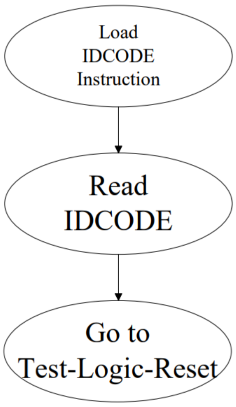
Appendix 2

Introduction to the operation process of JTAG interface. For the specific implementation of the C code of the following process, please refer to the application guide "AN03008 Compact Series CPLD Device JTAG Interface Configuration and Programming Application Guide" and the corresponding sample project.

In the following process, the number of continuous clock cycles in the Run Test Idle state is obtained from the fixed delay time when TCK is 50MHz, and the user can determine the number of clocks according to the actual TCK frequency. Users can determine the number of clocks to wait according to the actual TCK frequency. If TCK needs 100 cycles when it is 50MHz, then it needs 10 clock cycles when it is 5MHz.

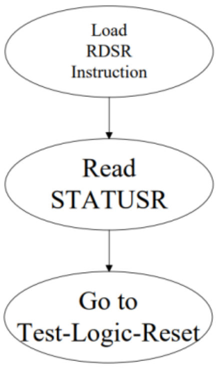
Chain sweep

Load IDCODE command to IR
Shift IDCODE to SHIFT-DR state, low bit first out
Enter Test logic Reset state



Read status register

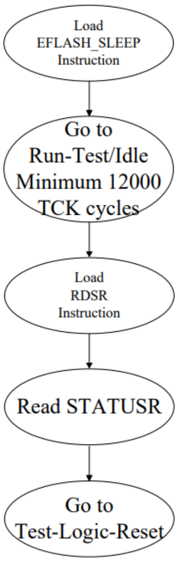
Load RDSR command to IR
Shift out the value of the status register to the SHIFT-DR state, low bit first out
Enter Test logic Reset state



Hibernate Embedded Flash

This operation can only be started when the embedded Flash is in an idle state (busy in the status register is 0).

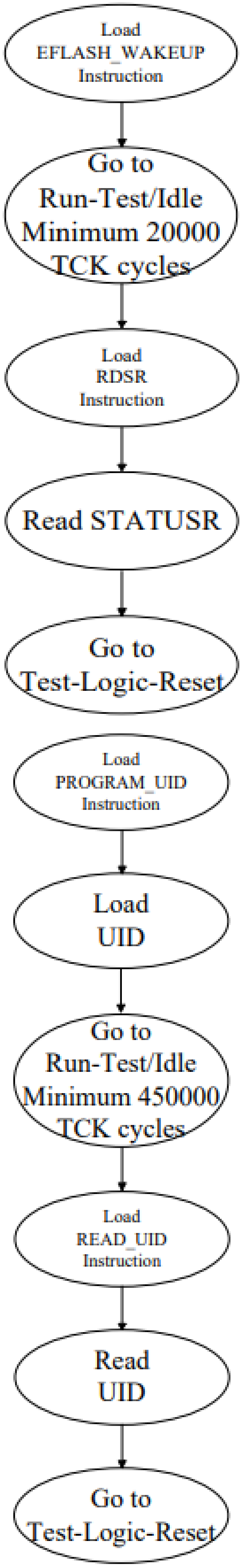
Load EFlash_SLEEP command to IR
Enter Run test Idle status, lasts at least 12000 clock cycles
Load RDSR command to IR
Shift out the content of the status register to the SHIFT-DR state, low bit first out
Enter Test logic Reset state



Wake up embedded Flash

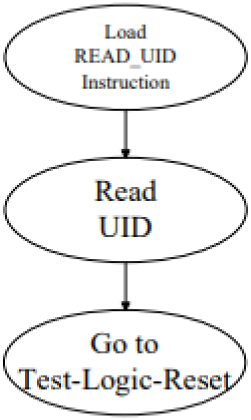
This operation can only be started when the embedded Flash is in an idle state (busy in the status register is 0).

Load EFlash_WAKEUP command to IR
Enter Run test Idle status, lasts at least 20000 clock cycles
Load RDSR command to IR
Shift out the content of the status register to the SHIFT-DR state, low bit first out
Enter Test logic Reset state


Load
EFLASH_WAKEUP
Instruction
Go to
Run-Test/Idle
Minimum 20000
TCK cycles
Load
RDSR
Instruction
Read STATUSR
Go to
Test-Logic-Reset
Load
PROGRAM_UID
Instruction
Load
UID
Go to
Run-Test/Idle
Minimum 450000
TCK cycles
Load
READ_UID
Instruction
Read
UID
Go to
Test-Logic-Reset

Read UID

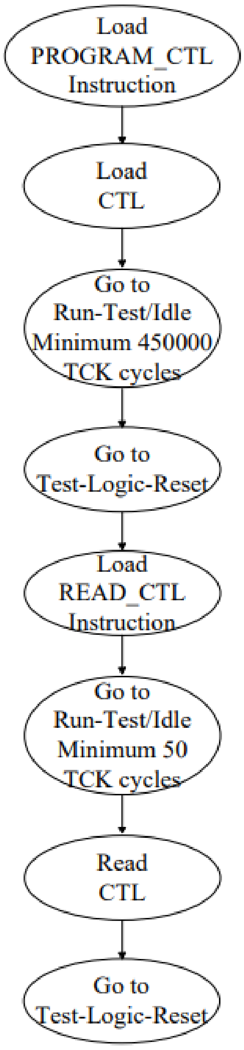
Load READ_UID command to IR
Move out UID in SHIFT-DR state, low byte first out
Enter Test logic Reset state



Programming Feature Control Bits

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

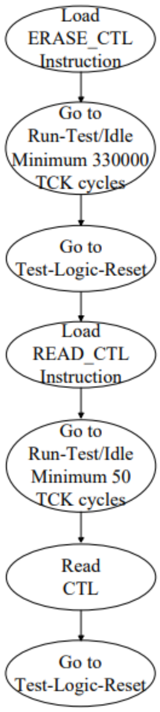
Load PROGRAM_CTL instruction into IR
To the SHIFT-DR state, shift into the feature control bit, the lower bit first
Enter the Run Test Idle state for a minimum of 450000 clock cycles
Enter Test Logic Reset state
Load READ_CTL command into IR
Enter the Run Test Idle state for a minimum of 50 clock cycles
To the SHIFT-DR state, shift out the feature control bit, the lower bit is first out
Enter Test Logic Reset state



Reset feature control bit

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

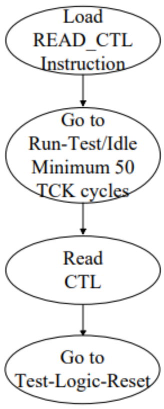
Load ERASE_CTL command into IR
Enter the Run Test Idle state for a minimum of 330000 clock cycles
Enter the Test Logic Reset state
Load READ_CTL command into IR
Enter the Run Test Idle state for a minimum of 50 clock cycles
To the SHIFT-DR state, shift out the feature control bit, the lower bit is first out
Enter the Test Logic Reset state



Read feature control bits

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

Load READ_CTL command into IR
Enter the Run Test Idle state for a minimum of 50 clock cycles
To the SHIFT-DR state, shift out the feature control bit, the lower bit is first out
Enter the Test Logic Reset state



Programming bit stream

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

Load WADR instruction into IR
Move to the SHIFT-DR state and enter the address of the first page of data in the bit stream in the embedded Flash, the lower bit is advanced Run Test Idle
Load PROGRAM instructions into IR
Shift in bitstream page 1 data Go to SHIFT-DR state, shift in 32 bits, high bit advanced, Run Test Idle until the end of page 1
Enter the Run Test Idle state for a minimum of 470000 clock cycles
Load READ instruction into IR
Enter the Run Test Idle state for a minimum of 50 clock cycles
Shift to the SHIFT-DR state to shift out the data of the first page of the bit stream, high bit first out
Load WADR instruction into IR
Shift to the SHIFT-DR state, the address of the data in the second page of the bit stream in the embedded Flash, low-order first
Load PROGRAM instructions into IR
Move to the SHIFT-DR state to move the data on the second page of the bit stream, each time 32 bits are shifted in, high-order first, Run Test Idle until the end of the second page
Enter the Run Test Idle state for a minimum of 470000 clock cycles
Load READ instruction into IR
Enter the Run Test Idle state for a minimum of 50 clock cycles
Shift to the SHIFT-DR state to shift out the data of the second page of the bit stream, high bit first out
……
Load WADR instruction into IR
Move to the SHIFT-DR state to shift the address of the data on page n of the bit stream in the embedded Flash, low order first
Load PROGRAM instructions into IR
Move to the SHIFT-DR state to shift the data of the nth page of the bit stream, shifting in 32 bits each time, high-order first, Run Test Idle
until the end of page n
Enter the Run Test Idle state for a minimum of 470000 clock cycles
load READ instruction into IR
Enter the Run Test Idle state for a minimum of 50 clock cycles
Shift to the SHIFT-DR state to shift out the nth page data of the bit stream, high bit first out
Enter Test Logic Reset state

Note: The value of n is equal to the size of the bitstream file to be loaded divided by the size of the embedded Flash per page

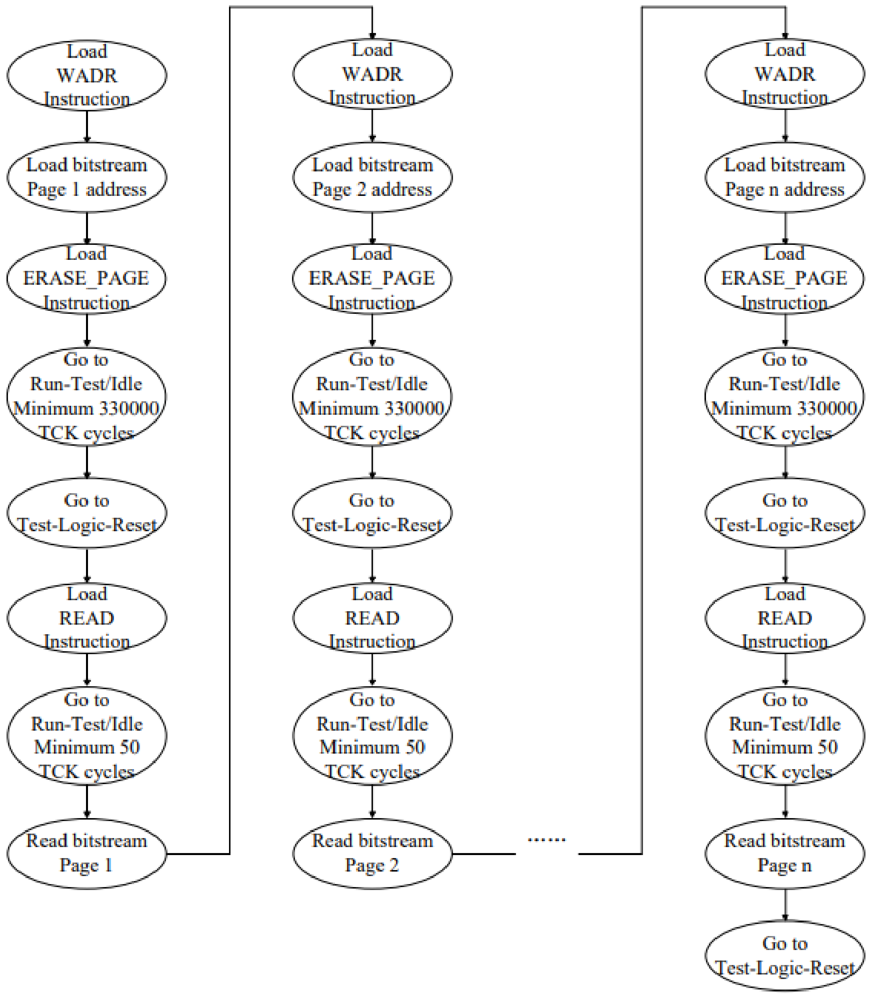


Erase bit stream

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

Load WADR instruction into IR
to SHIFT-DR state is shifted into bitstream 1 Page data in embedded Flash address in, LSB advanced
load ERASE_PAGE command to IR
Enter Run test Idle status, lasts at least 330000 clock cycle
Enter Test logic Reset state
load read command to IR
Enter Run test Idle status, lasts at least 50 clock cycle
to SHIFT-DR The state is shifted out of the first page data of the bit stream , the high bit is first out
load WADR command to IR

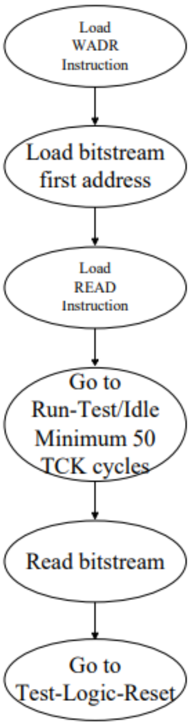
to SHIFT-DR state is shifted into bitstream 2 Page data in embedded Flash address in, LSB advanced
load ERASE_PAGE command to IR
Enter Run test Idle status, lasts at least 330000 clock cycles
Enter Test logic Reset state
load read command to IR
Enter Run test Idle status, lasts at least 50 clock cycles
Shift to the SHIFT-DR state to shift out the data of the second page of the bit stream, high bit first out
...
load WADR command to IR
Move to the SHIFT-DR state to shift the address of the data on page n of the bit stream in the embedded Flash, low order first
load ERASE_PAGE command to IR
Enter Run test Idle status, lasts at least 330000 clock cycles
Enter Test logic Reset state
load read command to IR
Enter Run test Idle status, lasts at least 50 clock cycles
to SHIFT-DR The status is shifted out of the nth page data of the bit stream, the high bit is first out
Enter Test logic Reset state



Read bitstream

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

Load WADR instruction into IR
Move to the SHIFT-DR state to shift the first address of the bit stream in the embedded Flash, the lower bit is advanced
Load READ instruction into IR
Enter the Run Test Idle state for a minimum of 50 clock cycles
Shift bit stream to SHIFT-DR state, MSB first
Enter the Test Logic Reset state



Programming User Flash

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

Load WADR instruction into IR
Move to the SHIFT-DR state to enter the address of the first page of user Flash data in the embedded Flash, low-order first
Load PROGRAM instructions into IR
Move to the SHIFT-DR state and move the data of page 1 of user Flash, each time 32 bits are moved in, high bit first, Run Test Idle until the end of page 1
Enter the Run Test Idle state for a minimum of 470000 clock cycles
Load READ instruction into IR
Enter the Run Test Idle state for a minimum of 50 clock cycles
Shift to the SHIFT-DR state to shift out the data of page 1 of the user Flash, high bit first out

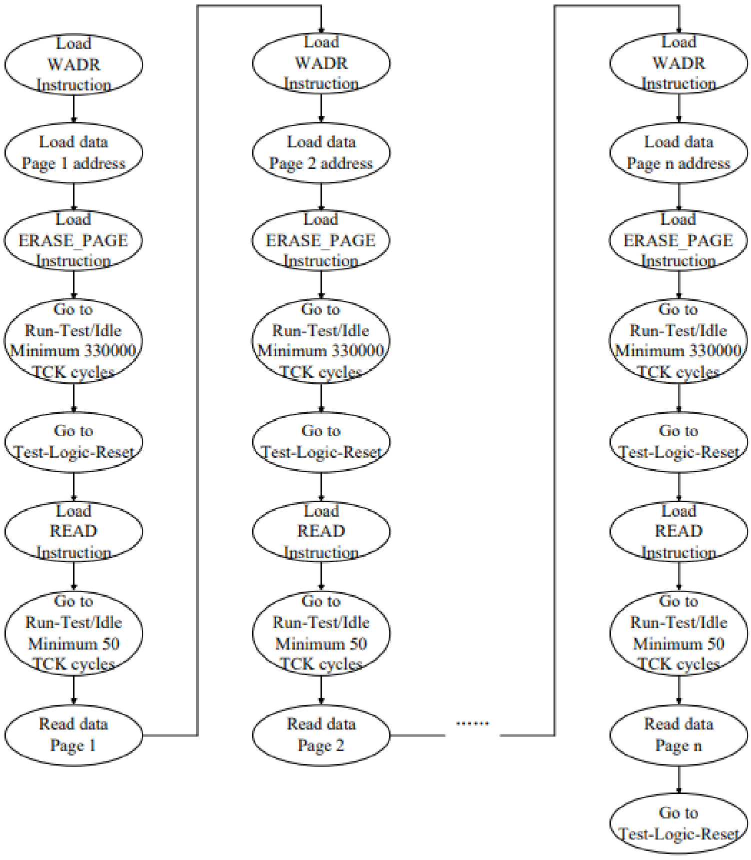
Load WADR instruction into IR
Move to the SHIFT-DR state to enter the address of the second page data in the user Flash in the embedded Flash, low-order first
Load PROGRAM instructions into IR
Move to the SHIFT-DR state to transfer the data of the second page of the user Flash, each time shifting 32 bits, high-order first, Run Test Idle until the end of the second page
Enter the Run Test Idle state for a minimum of 470000 clock cycles
Load READ instruction into IR
Enter the Run Test Idle state for a minimum of 50 clock cycles
Shift to the SHIFT-DR state to shift out the second page data of the user Flash, high bit first out
...
Load WADR instruction into IR
Move to the SHIFT-DR state and enter the address of the nth page data in the embedded Flash of the user Flash, the lower bit is advanced
Load PROGRAM instructions into IR
Move to the SHIFT-DR state to move the data of the nth page of the user Flash, each time 32 bits are moved in, high-order first, Run Test Idle until the end of the nth page
Enter the Run Test Idle state for a minimum of 470000 clock cycles
load READ instruction into IR
Enter the Run Test Idle state for a minimum of 50 clock cycles
Shift to the SHIFT-DR state to shift out the nth page data of the user Flash, high bit first out
Enter Test Logic Reset state



Erase User Flash

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

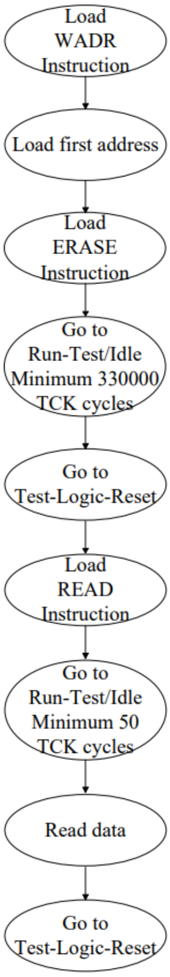
load WADR command to IR
to SHIFT-DR State moved into user Flash No. 1 Page data in embedded Flash address in, LSB advanced
load ERASE_PAGE command to IR
Enter Run test Idle status, lasts at least 330000 clock cycles
Enter Test logic Reset state
load read command to IR
Enter Run test Idle status, lasts at least 50 clock cycles
to SHIFT-DR State moved out of user Flash No. 1 Page data, MSB first out
load WADR command to IR
to SHIFT-DR State moved into user Flash No. 2 Page data in embedded Flash address in, LSB advanced
load ERASE_PAGE command to IR
Enter Run test Idle status, lasts at least 330000 clock cycles
Enter Test logic Reset state
load read command to IR
Enter Run test Idle status, lasts at least 50 clock cycles
to SHIFT-DR State moved out of user Flash No. 2 Page data, MSB first out
...
load WADR command to IR
to SHIFT-DR State moved into user Flash nth _ Page data in embedded Flash address in, LSB advanced
load ERASE_PAGE command to IR
Enter Run test Idle status, lasts at least 330000 clock cycles
Enter Test logic Reset state
load read command to IR
Enter Run test Idle status, lasts at least 50 clock cycles
to SHIFT-DR State moved out of user Flash nth _ Page data, MSB first out
Enter Test logic Reset state



Erase Bitstream and User Flash

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

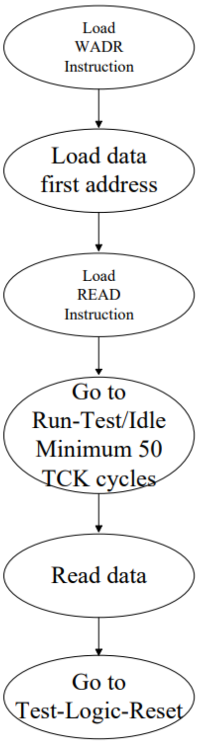
Load WADR instruction into IR
Shift into the SHIFT-DR state the bit stream and the first address of the user Flash data in the embedded Flash, low-order first
Load ERASE instruction into IR
Enter the Run Test Idle state for a minimum of 330000 clock cycles
Enter the Test Logic Reset state
Load READ instruction into IR
Enter the Run Test Idle state for a minimum of 50 clock cycles
Shift out bit stream and user Flash data in SHIFT-DR state, MSB first out
Enter the Test Logic Reset state



Read User Flash

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

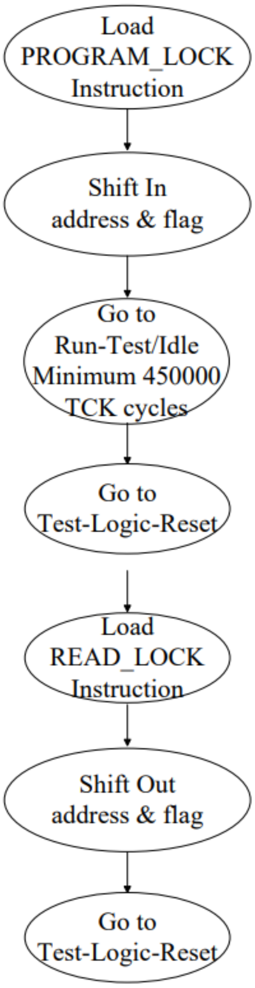
Load WADR instruction into IR
Move to the first address of the user Flash data in the embedded Flash in the SHIFT-DR state, the lower bit is advanced
Load READ instruction into IR
Enter the Run Test Idle state for a minimum of 50 clock cycles
Shift out user Flash data in SHIFT-DR state, high bit first out
Enter the Test Logic Reset state



Lock Embedded Flash

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

Load PROGRAM_LOCK instruction into IR
Shift into the SHIFT-DR state the last page address of the bit stream and the embedded Flash lock flag bit, the lower bit is advanced
Enter the Run Test Idle state for a minimum of 450000 clock cycles
Enter the Test Logic Reset state
Load READ_LOCK instruction into IR
Shift out bit stream tail page address and embedded Flash lock flag bit in SHIFT-DR state, LSB first out
Enter the Test Logic Reset state

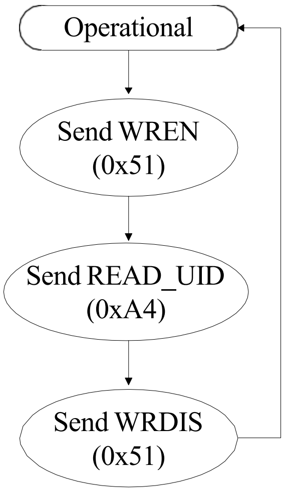

Load
PROGRAM_LOCK
Instruction
Shift In
address & flag
Go to
Run-Test/Idle
Minimum 450000
TCK cycles
Go to
Test-Logic-Reset
Load
READ_LOCK
Instruction
Shift Out
address & flag
Go to
Test-Logic-Reset

Appendix 3

The SPI/Slave I²C interface operation flow is described. For the specific implementation of the C code of the following process, please refer to the application guide "AN03009 Application Guide for Configuration and Programming of Compact Series CPLD Devices from SPI Interface" and "AN03010 Application Guide for Configuration and Programming of Compact Series CPLD Devices from I²C Interface", and corresponding examples project.

Read UID

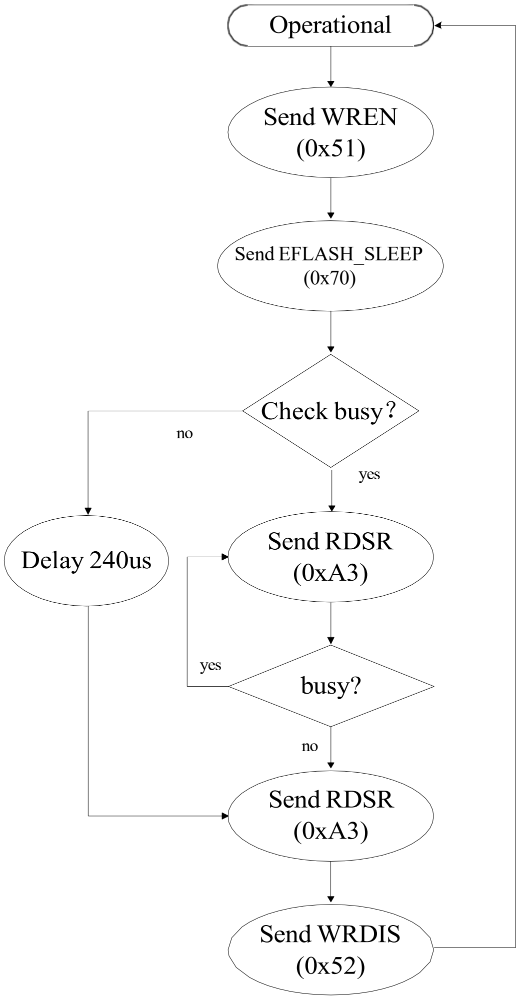
write WREN instruction
write RDUID instruction
write WRDIS instruction



Hibernate Embedded Flash

This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

write WREN instruction
write EFlash_SLEEP instruction
write RDSR command, read register STATUSR cyclically until a busy is detected for low
write RDSR instruction
write WRDIS instruction

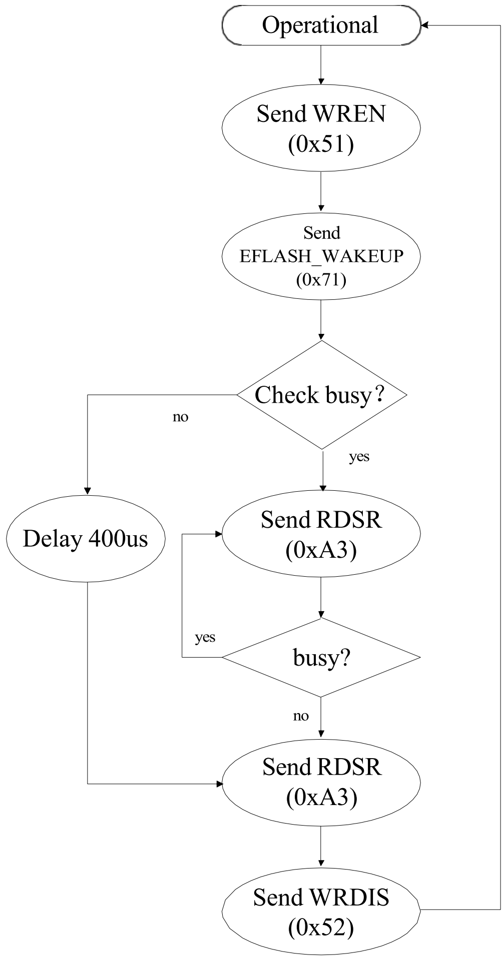


Wake up embedded Flash

This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

Table 36 Wake up embedded Flash

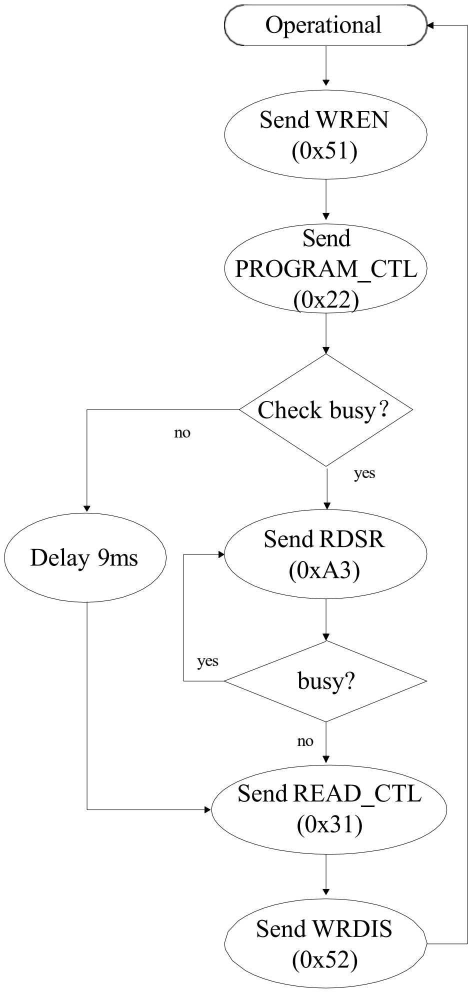
write WREN instruction
Write EFlash_WAKEUP instruction
write RDSR command, read register STATUSR cyclically until a busy is detected for low
write RDSR instruction
write WRDIS instruction



Programming Feature Control Bits

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

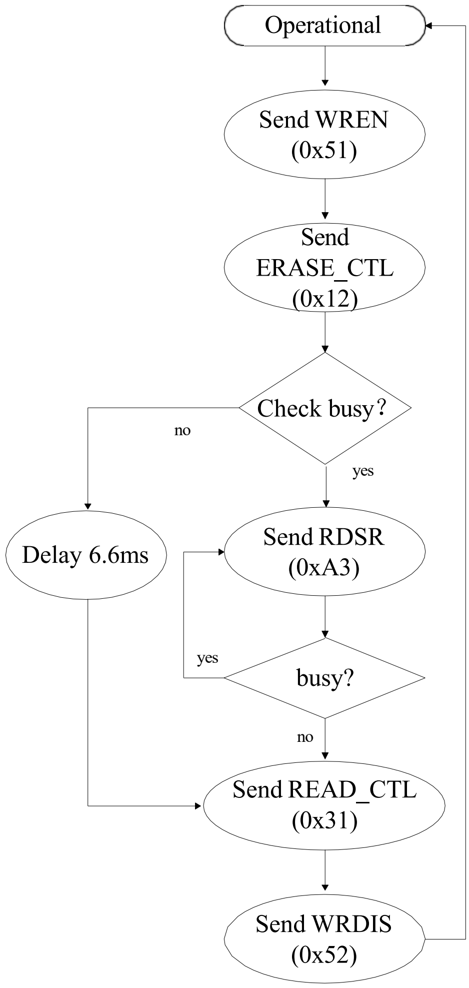
Write WREN instruction
Write PROGRAM_CTL instruction
Write RDSR instruction, read STATUSR register circularly, until busy is detected as low
Write READ_CTL instruction
Write WRDIS command



Reset feature control bit

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

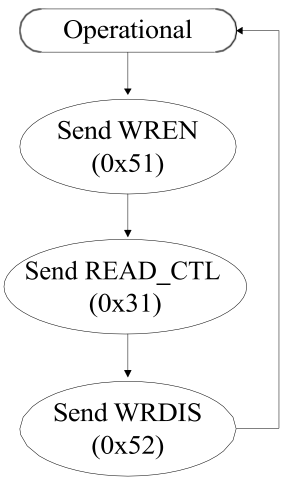
Write WREN instruction
Write ERASE_CTL instruction
Write RDSR instruction, read STATUSR register circularly, until busy is detected as low
Write READ_CTL instruction
Write WRDIS command



Read feature control bits

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

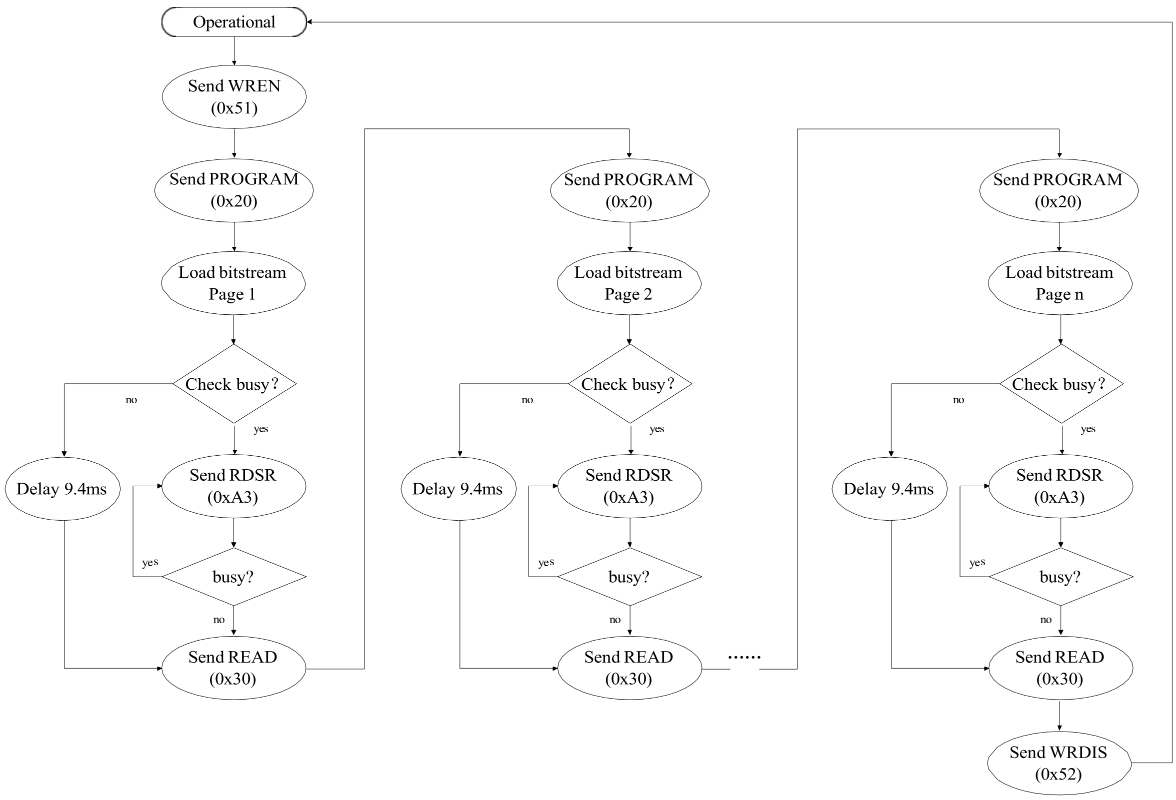
Write WREN instruction
Write READ_CTL instruction
Write WRDIS command



Programming bit stream

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

Write WREN instruction
Write PROGRAM command, program bit stream page 1 data
Write RDSR instruction, read STATUSR register circularly, until busy is detected as low
Write the READ instruction to read the data of page 1 of the bit stream
Write PROGRAM command, program bit stream page 2 data
Write RDSR instruction, read STATUSR register circularly, until busy is detected as low
Write the READ command to read the data of the second page of the bit stream
……
Write PROGRAM command, program bit stream page n data
Write RDSR instruction, read STATUSR register circularly, until busy is detected as low
Write the READ command to read the nth page data of the bit stream
Write WRDIS command

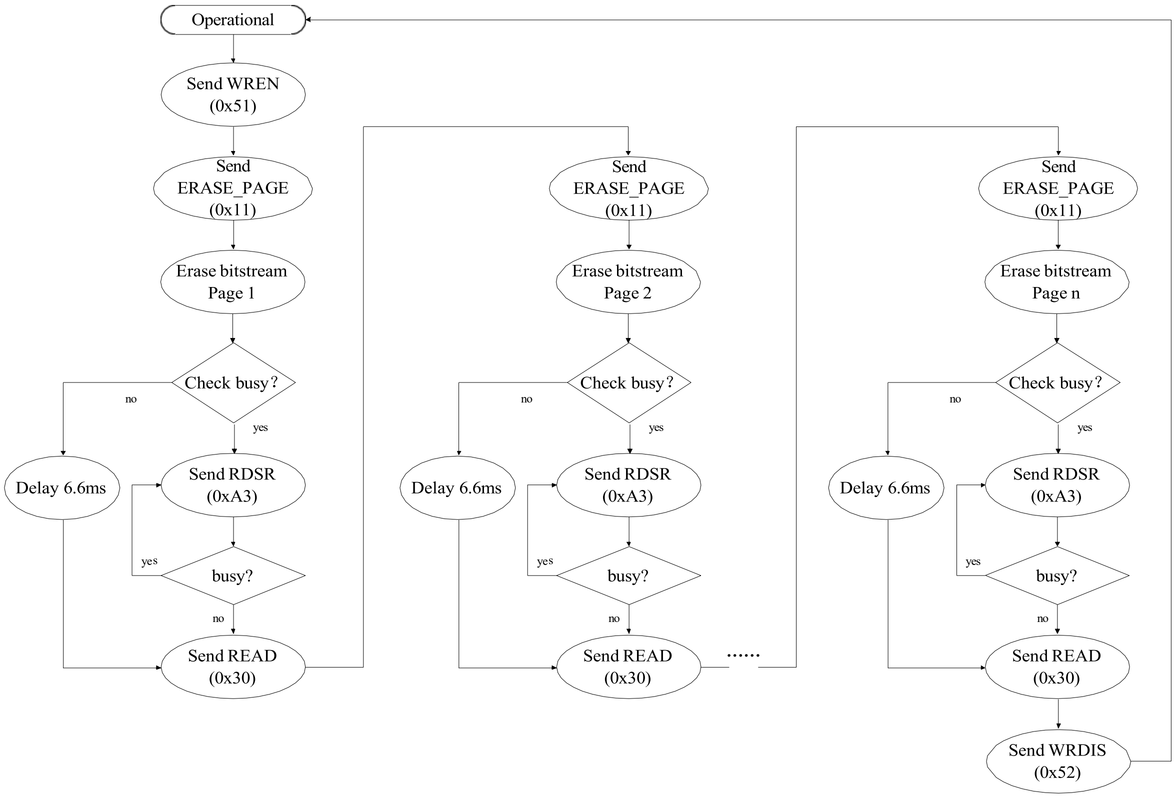


Erase bit stream

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

Write WREN instruction
Write the ERASE_PAGE command to erase the data on page 1 of the bit stream
Write RDSR instruction, read STATUSR register circularly, until busy is detected as low

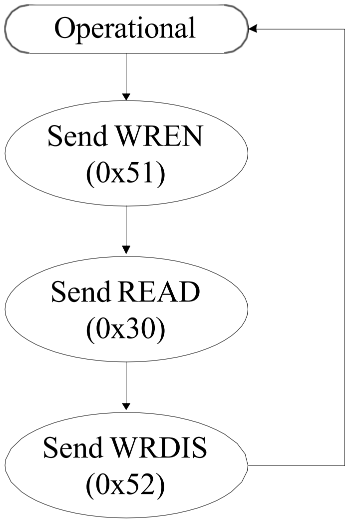
write read instruction, read bitstream 1st page data
write ERASE_PAGE instruction to erase the data on page 2 of the bitstream
write RDSR command, read STATUSR cyclically register until a busy is detected for low
write read instruction, read bitstream 2 page data
...
write ERASE_PAGE command, erase the data on the nth page of the bit stream
write RDSR command, read STATUSR cyclically register until a busy is detected for low
write read instruction, read bitstream nth page data
write WRDIS instruction



Read bitstream

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

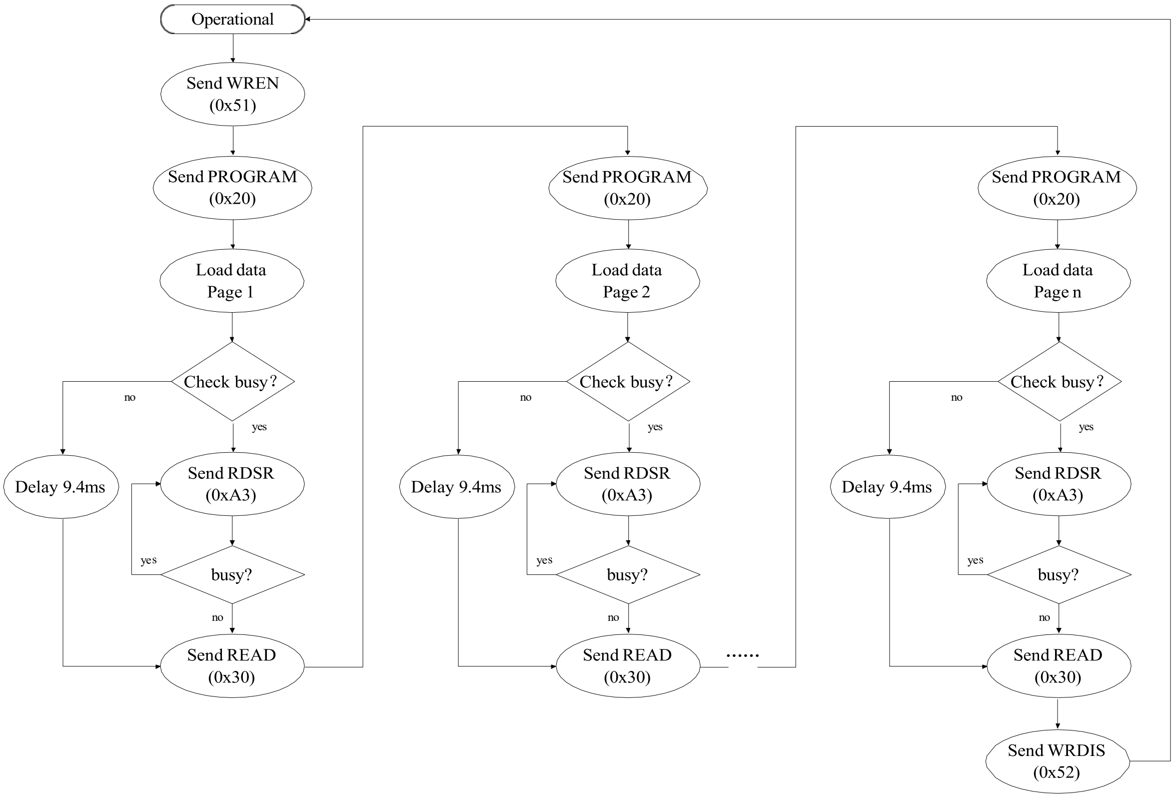
Write WREN instruction
Write READ instruction, read bit stream
Write WRDIS command



Programming User Flash

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

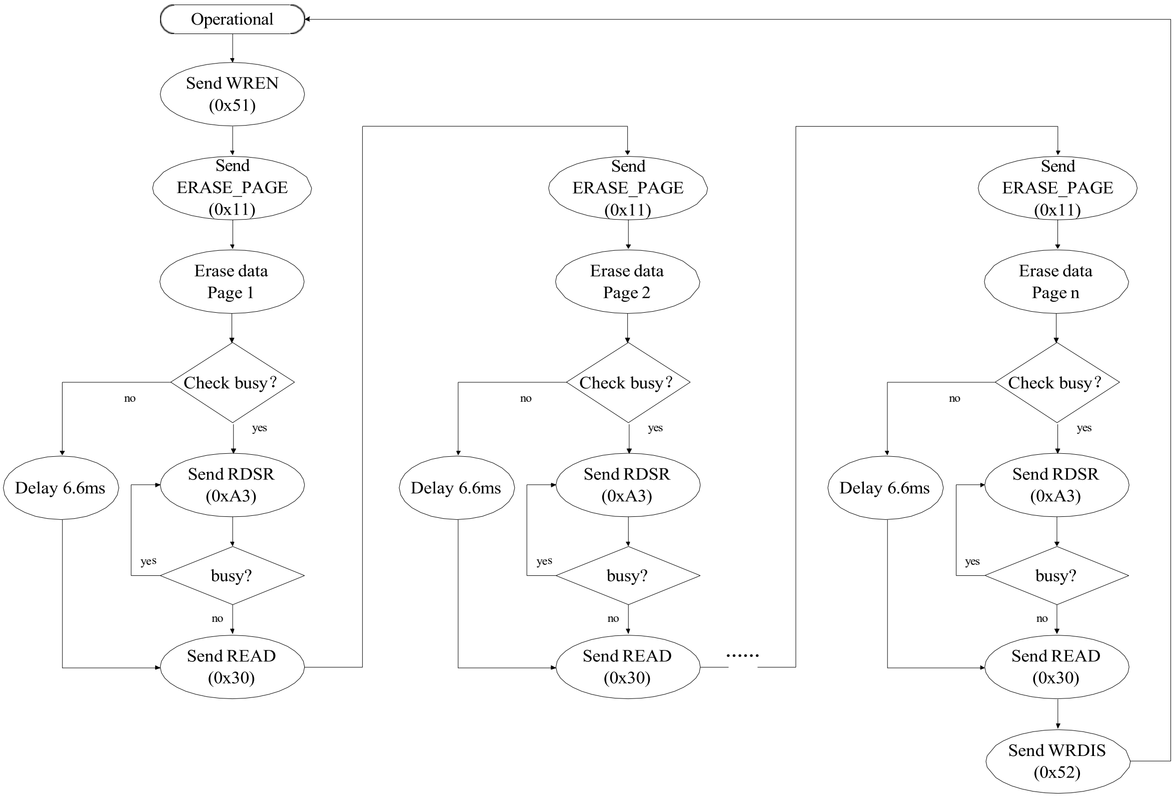
Write WREN instruction
Write PROGRAM command, program user Flash page 1 data
Write RDSR instruction, read STATUSR register circularly, until busy is detected as low
Write the READ command to read the first page data of user Flash
Write PROGRAM command, program user Flash page 2 data
Write RDSR instruction, read STATUSR register circularly, until busy is detected as low
Write the READ command to read the second page data of the user Flash
……
Write PROGRAM command, program user Flash page n data
Write RDSR instruction, read STATUSR register circularly, until busy is detected as low
Write the READ command to read the nth page data of the user Flash
Write WRDIS command



Erase User Flash

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

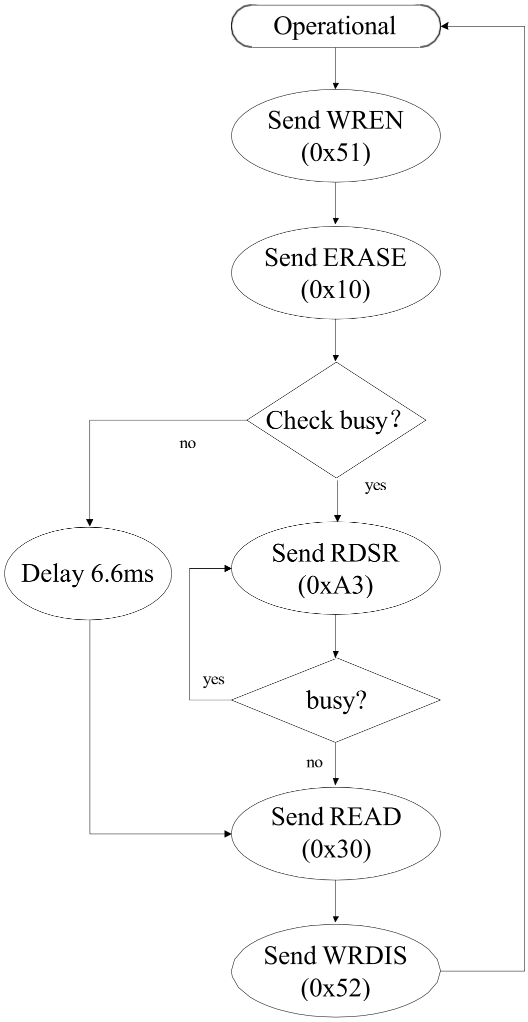
Write WREN instruction
Write the ERASE_PAGE command to erase the data on page 1 of user Flash
Write RDSR instruction, read STATUSR register circularly, until busy is detected as low
Write the READ command to read the first page data of user Flash
Write the ERASE_PAGE command to erase the data on page 2 of user Flash
Write RDSR instruction, read STATUSR register circularly, until busy is detected as low
Write the READ command to read the second page data of the user Flash
......
Write the ERASE_PAGE command to erase the nth page data of user Flash
Write RDSR instruction, read STATUSR register circularly, until busy is detected as low
Write the READ command to read the nth page data of the user Flash
Write WRDIS command



Erase Bitstream and User Flash

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

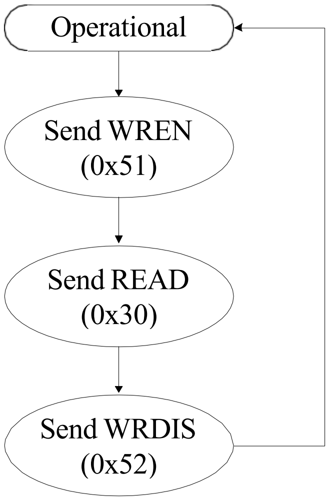
Write WREN instruction
Write ERASE instruction
Write RDSR instruction, read STATUSR register circularly, until busy is detected as low
Write READ instruction to read bit stream and user Flash data
Write WRDIS command



Read User Flash

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

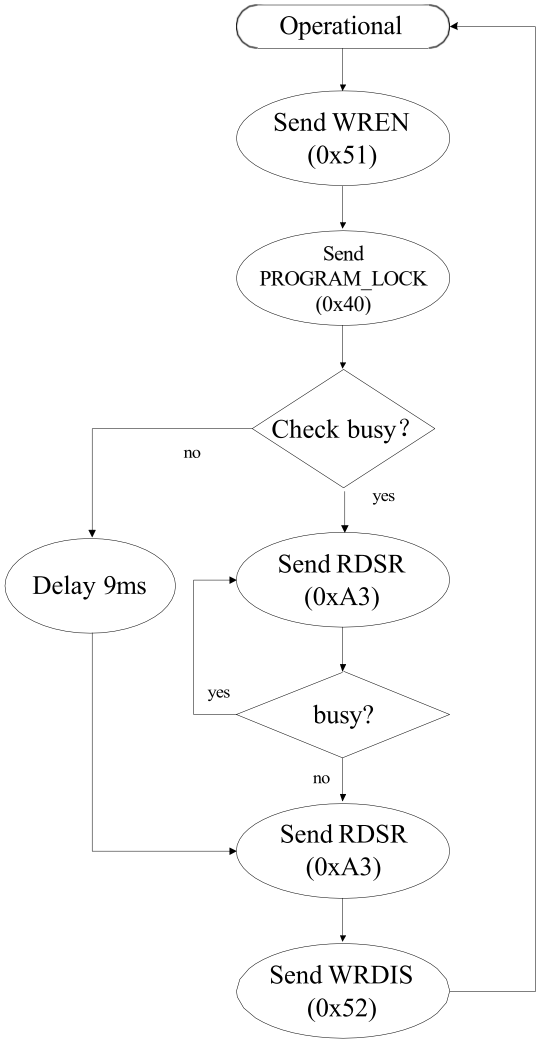
Write WREN instruction
Write READ command to read user Flash data
Write WRDIS command



Lock Embedded Flash

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in the idle state (the busy bit of the status register is 0).

Write WREN instruction
Write PROGRAM_LOCK instruction
Write RDSR instruction, read STATUSR register circularly, until busy is detected as low
Write RDSR instruction
Write WRDIS command

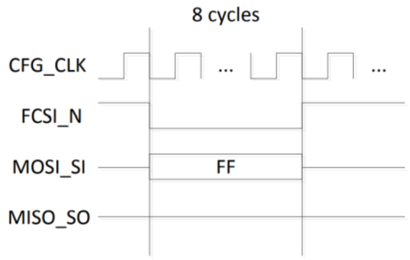

Operational
Send WREN
(0x51)
Send
PROGRAM_LOCK
(0x40)
Check busy?
no
yes
Delay 9ms
Send RDSR
(0xA3)
yes
busy?
no
Send RDSR
(0xA3)
Send WRDIS
(0x52)

Appendix 4

Instructions from the SPI interface. When writing a command, write the high bit of the command first; when writing an address, write the high bit of the address first. All the following instructions need to pull FCSI_N high before starting and after the instruction is completed, and give at least 8 clocks.

NOP

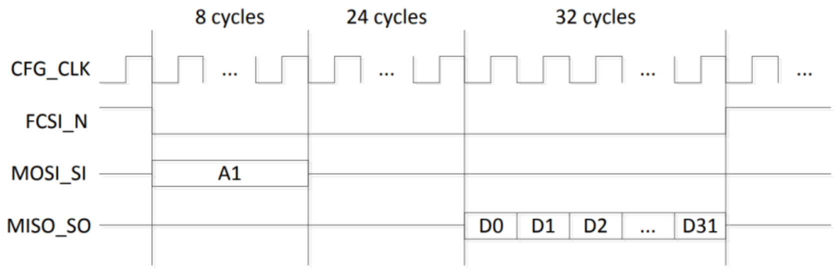
No operation instruction.



RDID

Read IDCODE instruction.

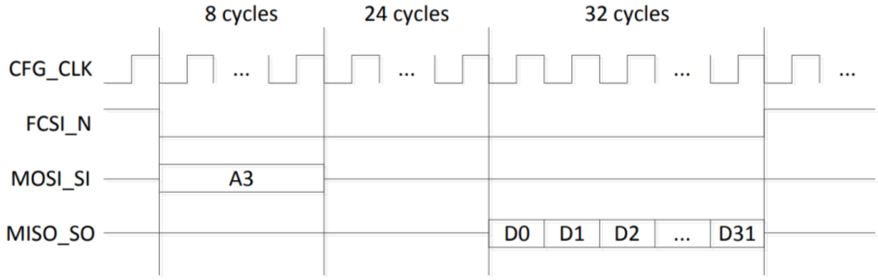
Used to read back the IDCODE of the CPLD device, LSB first out.



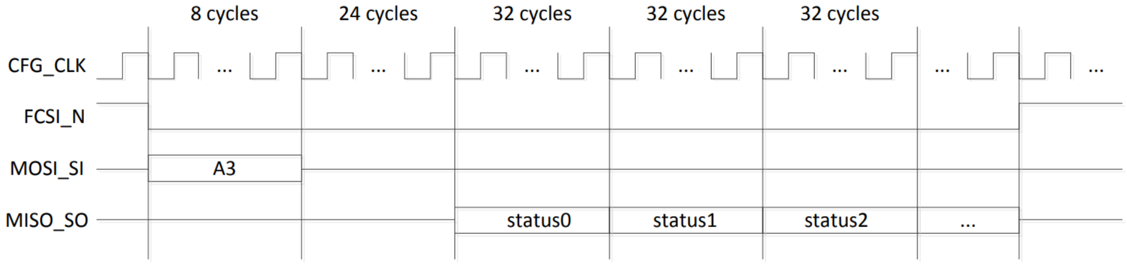
RDSR

Read status register instruction.

Used to read back the status register, LSB first out.



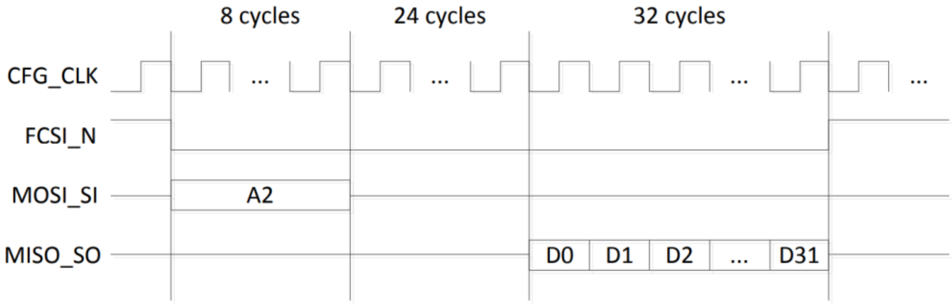
The status register can be read back continuously.



RDUSER

Read USERCODE instruction.

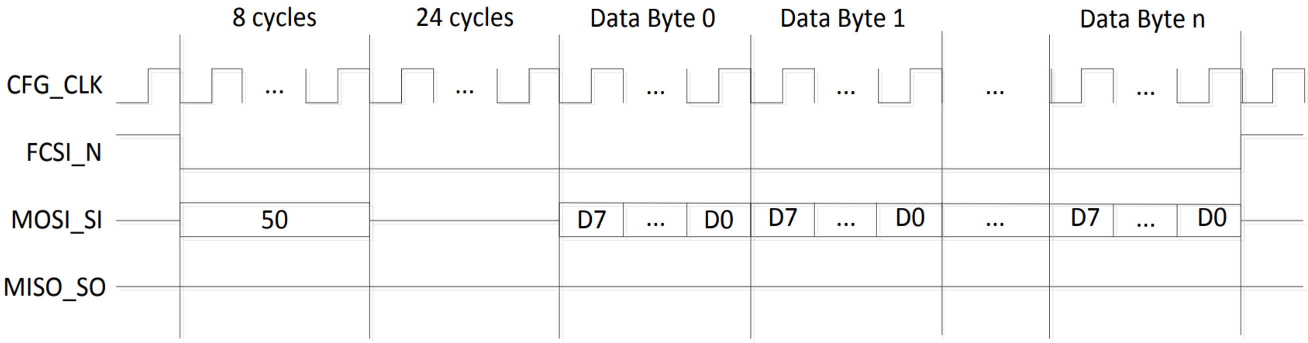
Used to read back the USERCODE of the CPLD device, LSB first out.



CFG

Configuration directives.

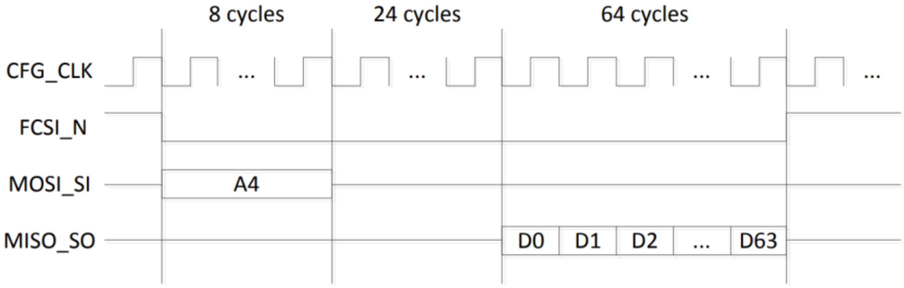
Used to load bit stream, configure CPLD device.



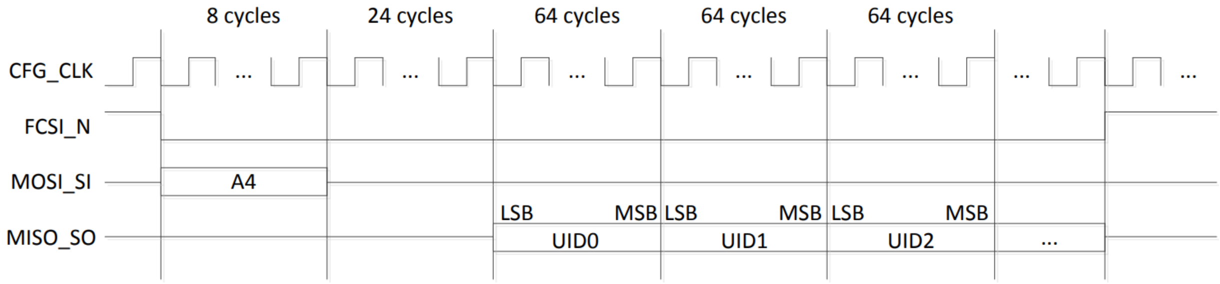
RDUID

Read UID instruction.

It is used to read back the UID of the CPLD device, LSB first out.



The UID of the CPLD device can be continuously read back.



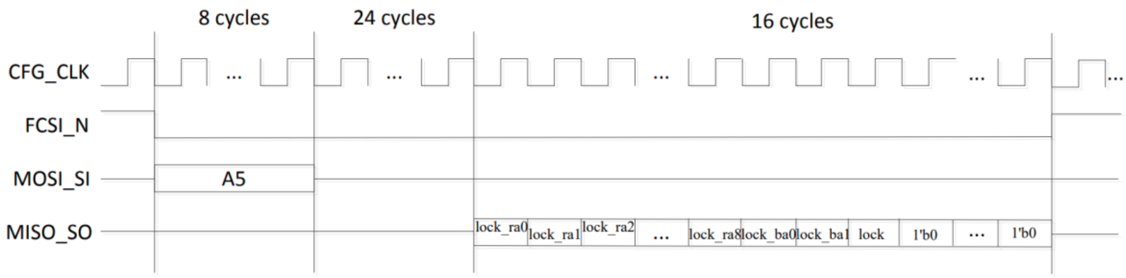
RDLOCK

Read embedded Flash lock information command.

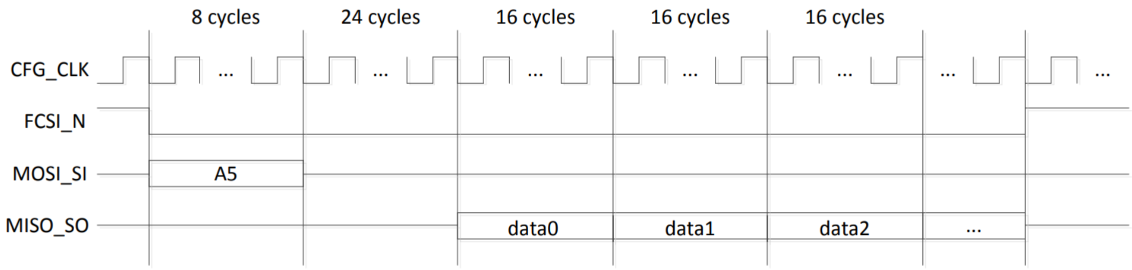
Used to read back the last page address of the bit stream and the embedded Flash lock flag, LSB first out. Move out the page address lock_ra[0:8] of the last page of the bit stream, the heap address lock_ba[0:1], the lock flag lock and 4’d0 in sequence.

For 1K devices, lock_ba[1:0] is a fixed value of 2’b11.

For 2K devices, lock_ba[1:0] is the reserved heap address.



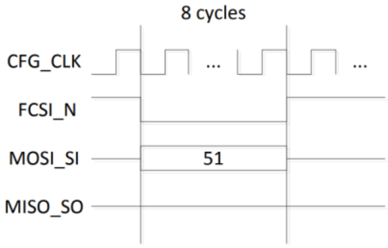
The lock information of the embedded Flash can be continuously read back.



WREN

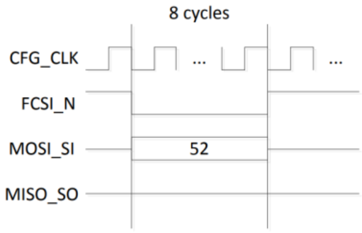
Write enable command.

Used before configuration directives and various programming directives.



WRDIS

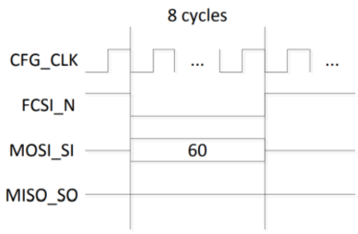
Write inhibit instruction.



RESET

Reset command.

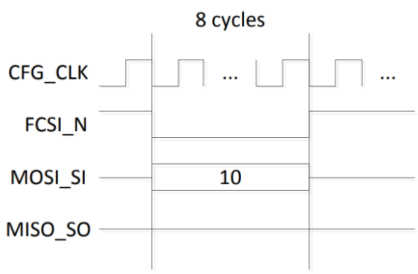
Used to reset parts of the configuration control system other than JTAG.



ERASE

Erase command.

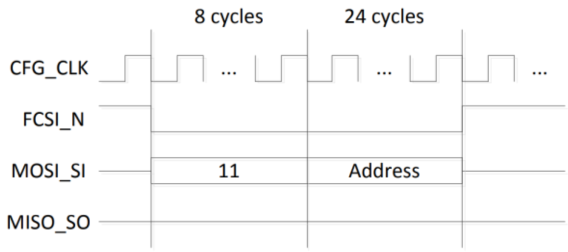
Used to erase embedded Flash all regular pages and clear embedded Flash Lock flag.



ERASE_PAGE

Page erase instruction.

Normal page erase for embedded Flash.



ADDRESS	NAME	DESCRIPTION
[23:19]	reserved	Need to be set to 5'b00000
[18:17]	Ba[1:0]	Heap address
[16:8]	Ra[8:0]	Page address
[7:0]	reserved	Needs to be set to 8'b00000000

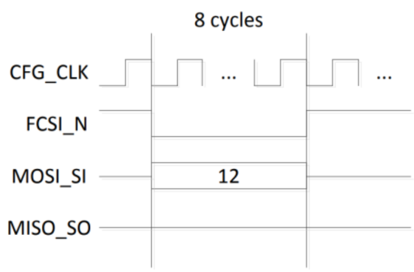
for 1K devices, Ba[1:0] are reserved bits.

for 2K device, Ba[1:0] is the reserved heap address and needs to be set to 2'b11 .

ERASE_CTL

Controls the erase command.

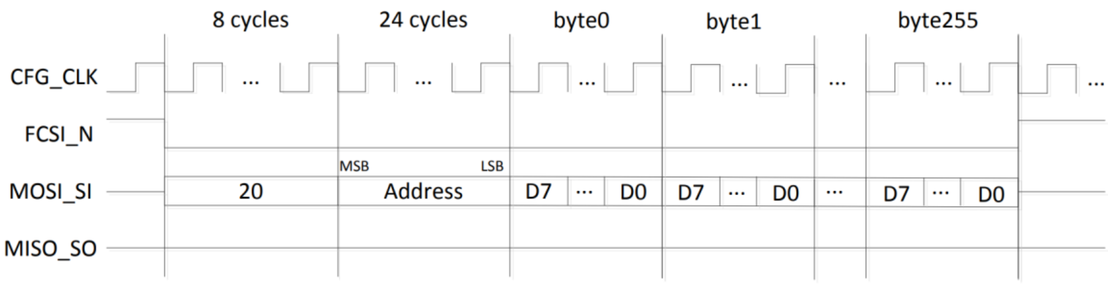
Feature control page for erasing embedded Flash.



PROGRAM

Programming instructions.

Used to program embedded Flash general pages.



ADDRESS	NAME	DESCRIPTION
[23:19]	reserve	Need to be set to 5'b00000

[18:17]	Ba[1:0]	Heap address
[16:8]	Ra[8:0]	Page address
[7:0]	reserve	Needs to be set to 8'b00000000

for 1K devices, Ba[1:0] are reserved bits.

for 2K device, Ba[1:0] is the reserved heap address and needs to be set to 2'b11 .

PROGRAM_CTL

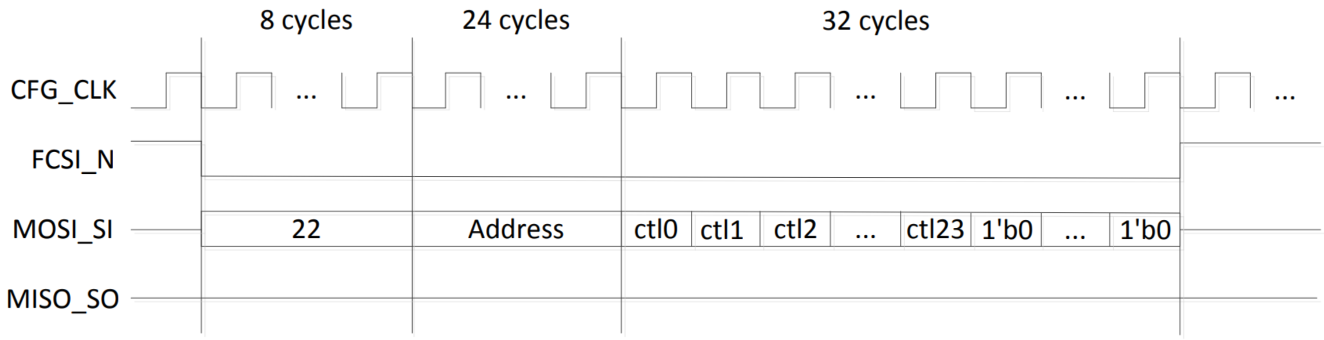
Control programming instructions.

Feature control page for programming the embedded Flash.

For 2K devices, data is shifted into ctl[0:23] followed by 8’d0.

For 1K/4K/7K devices, data is sequentially shifted into ctl[0:31].

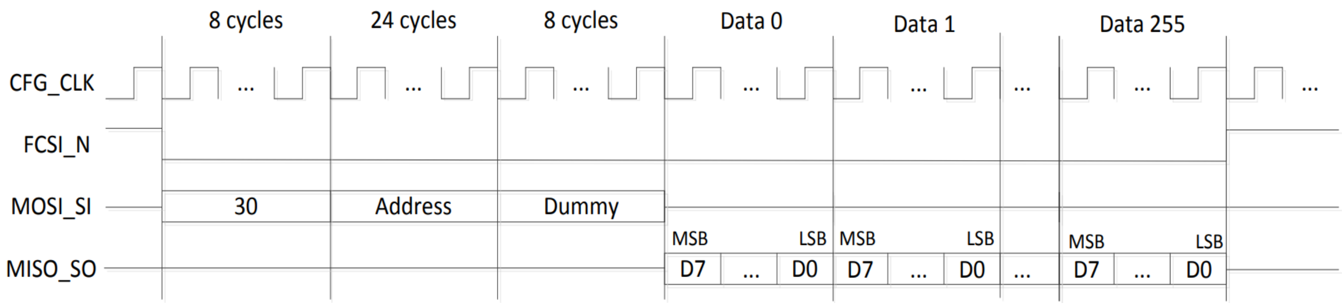
24-bit addresses are reserved.



READ

Read command.

Used to read embedded Flash normal page.



ADDRESS	NAME	DESCRIPTION
[23:19]	reserve	Need to be set to 5'b00000
[18:17]	Ba[1:0]	Heap address
[16:8]	Ra[8:0]	Page address
[7:6]	reserve	Need to be set to 2'b00
[5:0]	Ca[5:0]	32 Bit data page offset address

for 1K devices, Ba[1:0] are reserved bits.

for 2K device, Ba[1:0] is the reserved heap address and needs to be set to 2'b11 .

READ_CTL

Control read command.

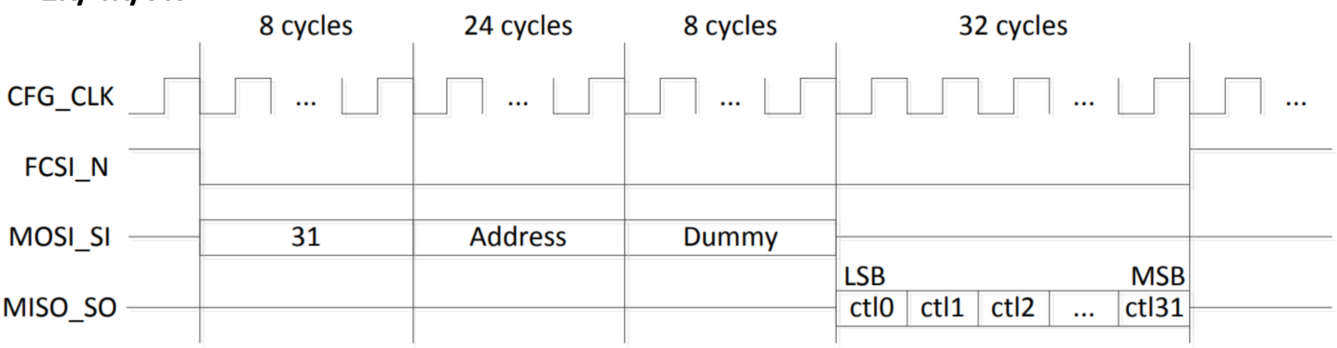
The feature used to read embedded Flash controls the content of the page.

For 2K devices, data is shifted out of ctl[0:23] followed by 8’d0.

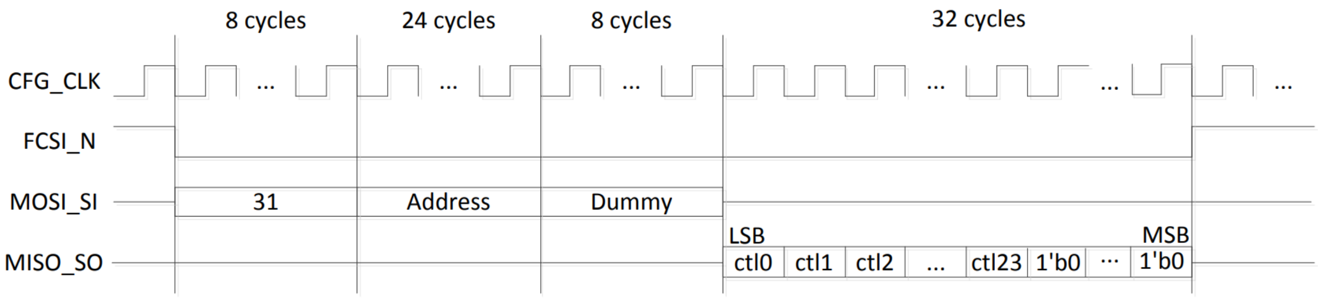
For 1K/4K/7K devices, data is shifted out of ctl[0:31] sequentially.

24-bit addresses are reserved.

1K/4K/7K



2K



PROGRAM_LOCK

Lock instruction.

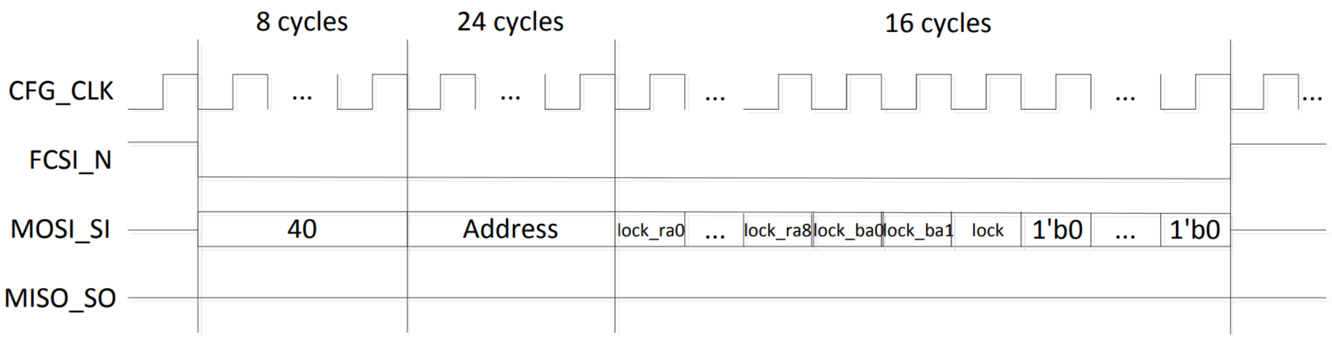
Used to inhibit programming and reading operations on the bit stream.

The data is sequentially shifted into the page address lock_ra[0:8] of the last page of the bitstream, the heap address lock_ba[0:1], the lock flag lock and 4’d0.

For 1K devices, lock_ba[1:0] are reserved bits.

For 2K devices, lock_ba[1:0] is the reserved heap address and needs to be set to 2’b11.

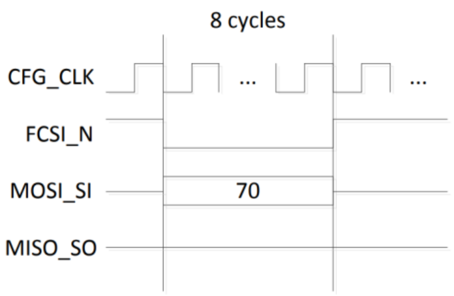
24-bit addresses are reserved.



EFlash_SLEEP

Embedded Flash sleep command.

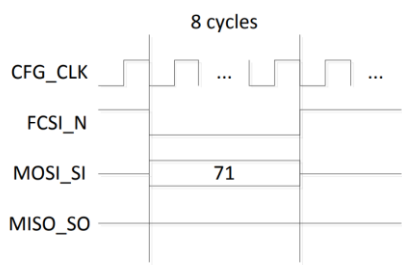
Used to hibernate embedded Flash.



EFlash_WAKEUP

Embedded Flash wake-up command.

Used to wake up the embedded Flash.



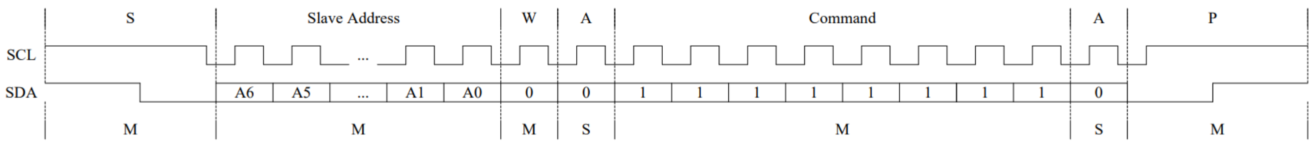
Appendix 5

I^{2}C Interface instruction timing introduction.

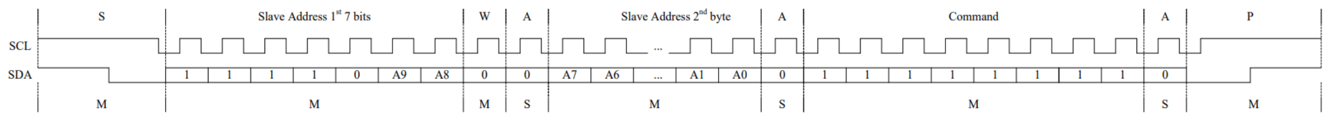
NOP

No operation instruction.

7 bit addressing



10 bit addressing

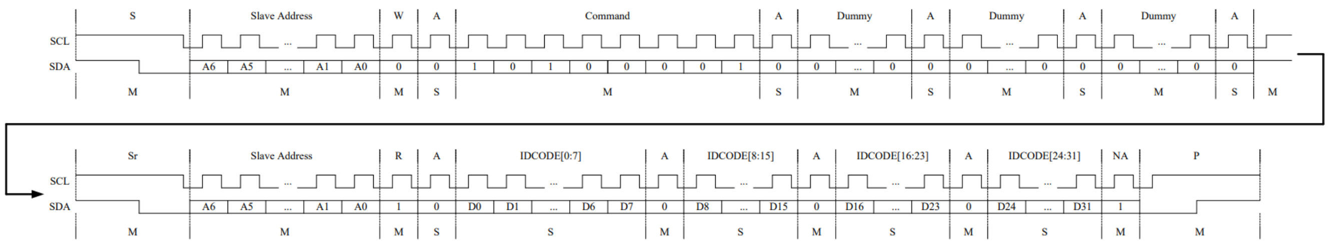


RDID

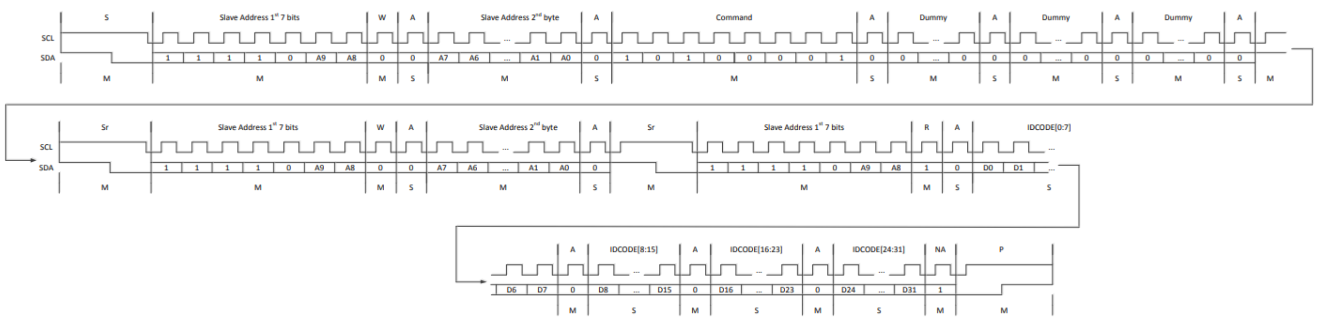
Read IDCODE instruction.

It is used to read back the IDCODE of the CPLD device, LSB first out.

7 bit addressing



10 bit addressing

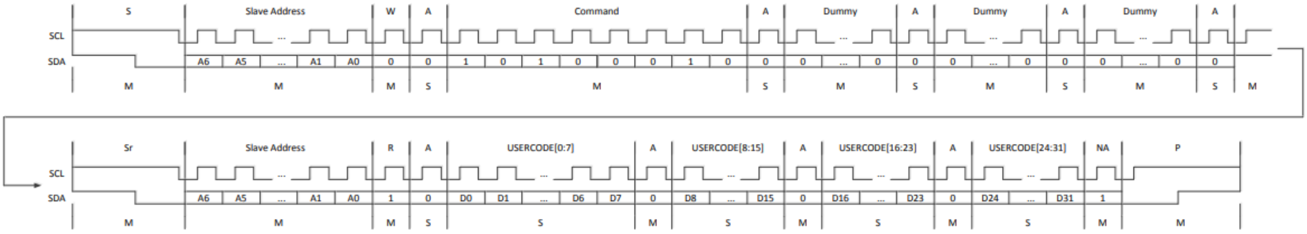


RD USER

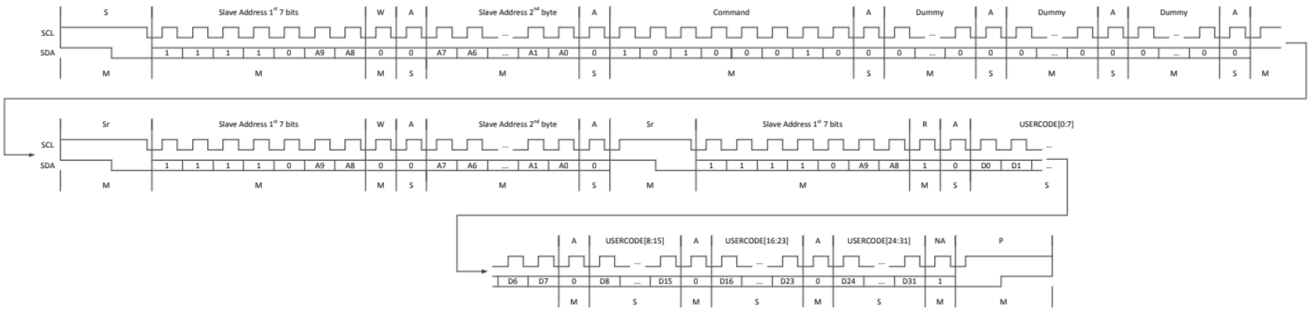
Read the USERCODE directive.

Used to read back the USERCODE of the CPLD device, LSB first out.

7 bit addressing



10 bit addressing

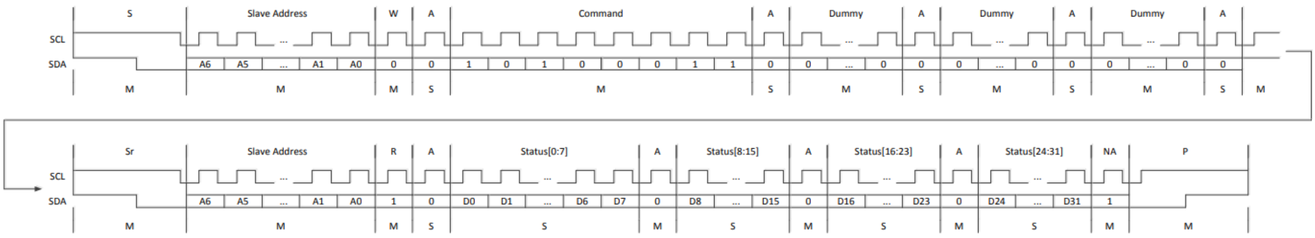


RDSR

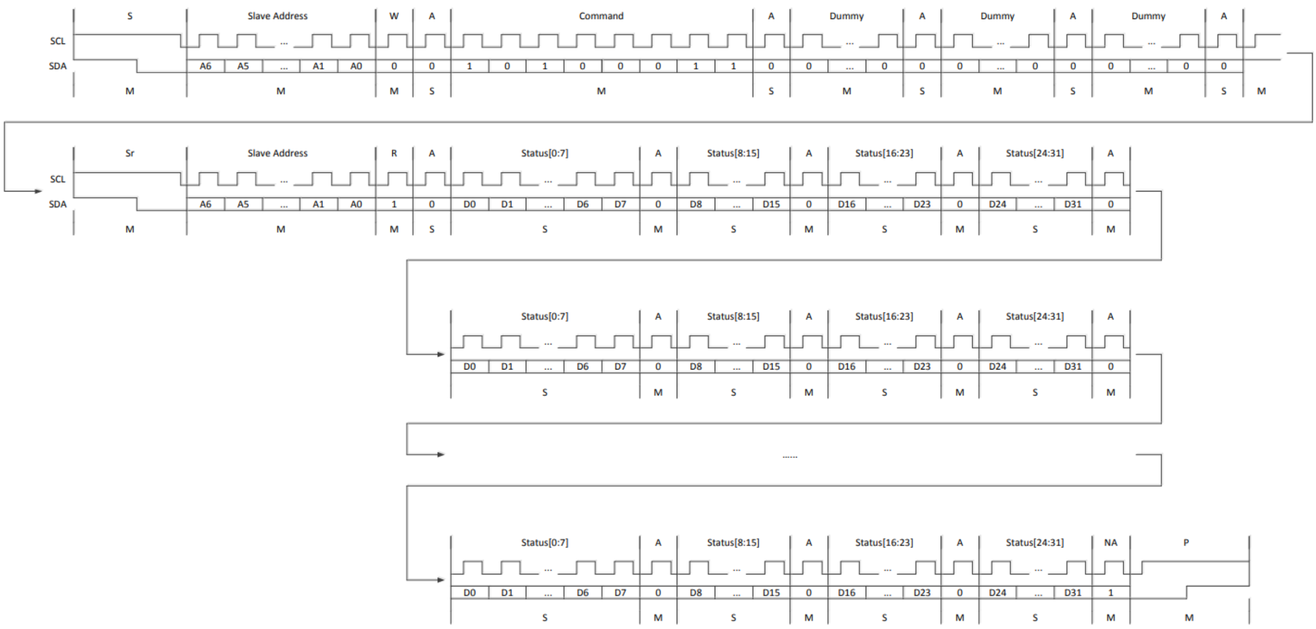
Read status register instruction.

Used to read back the status register, LSB first out.

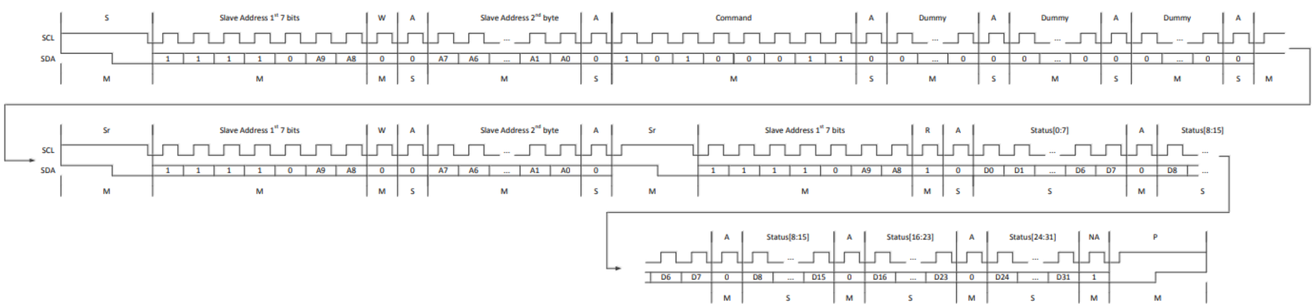
7 bit addressing



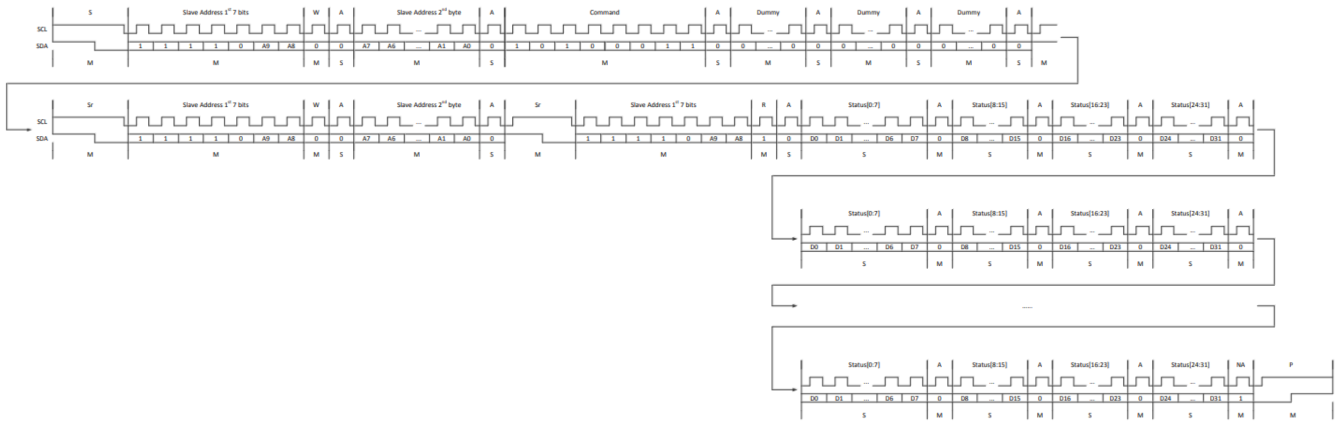
The status register can be read back continuously.



10 bit addressing



The status register can be read back continuously.

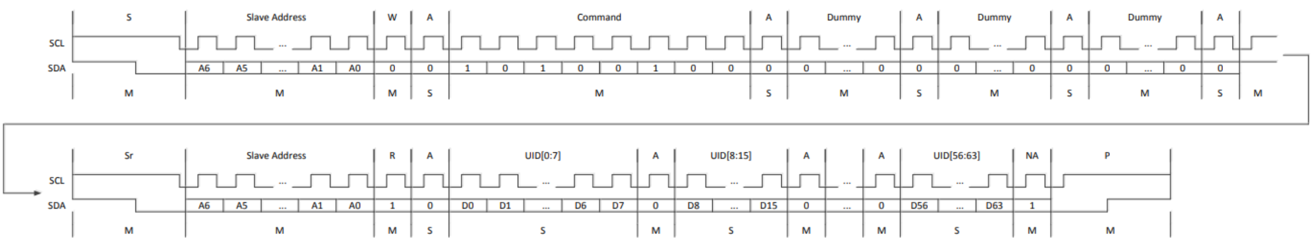


RDUID

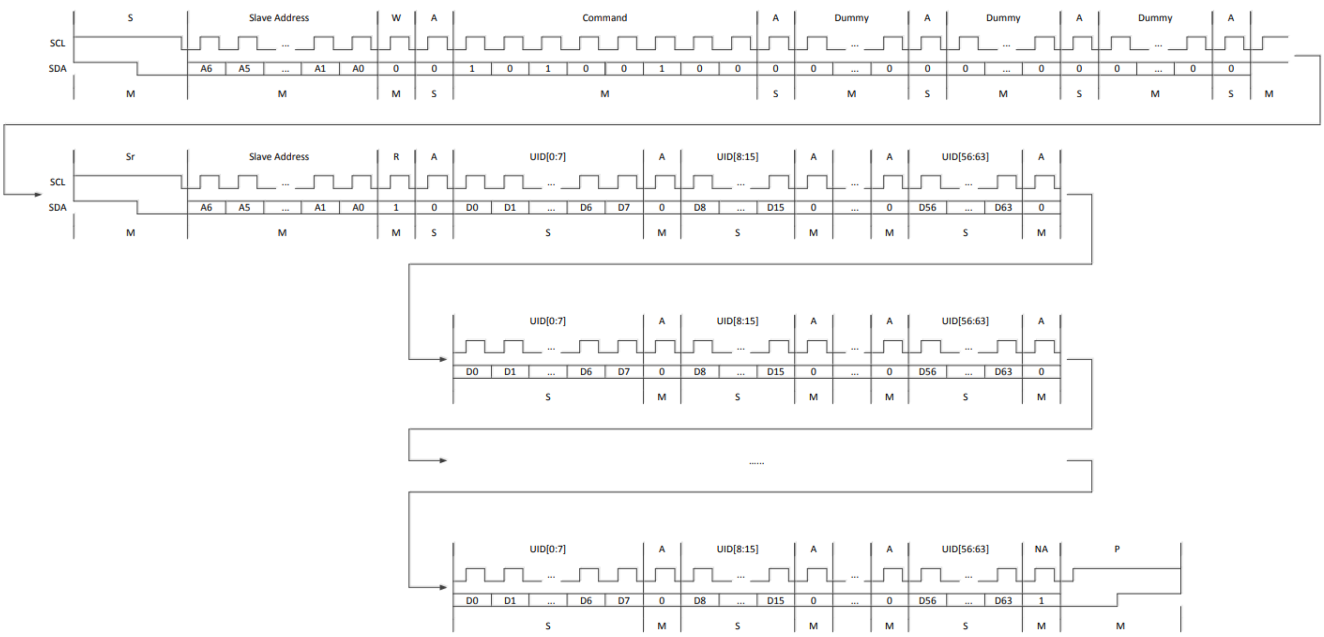
Read UID command.

Used to read back the UID of the CPLD device, LSB first out.

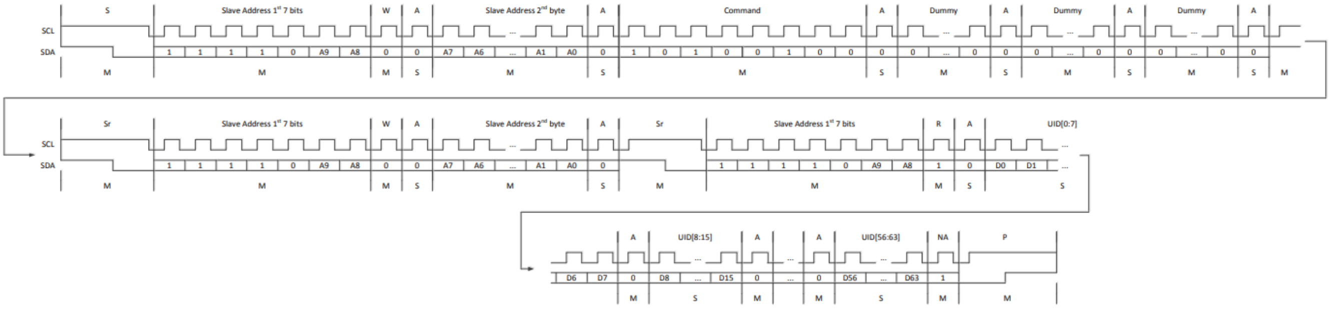
7 bit addressing



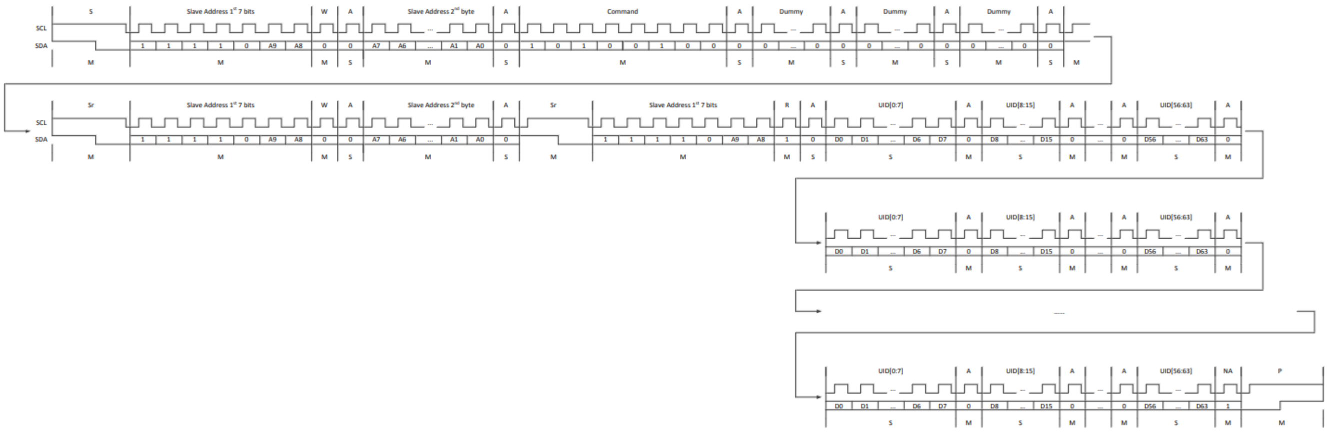
The UID of the CPLD device can be continuously read back.



10 bit addressing



The UID of the CPLD device can be continuously read back.



RDLOCK

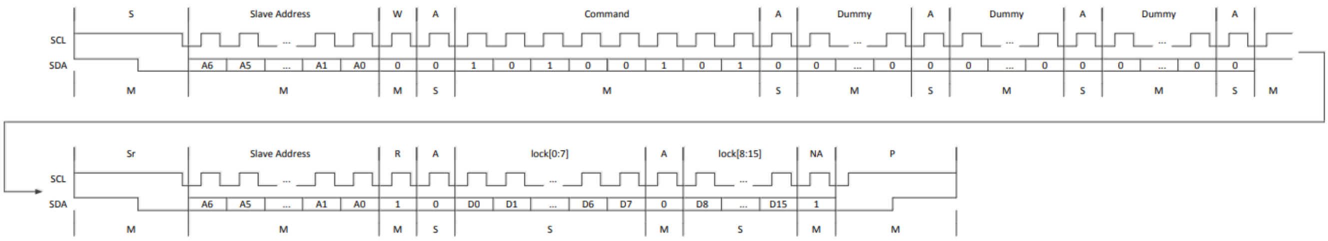
Read embedded Flash lock information command.

Used to read back the last page address of the bit stream and the embedded Flash lock flag, LSB first out. Move out the page address lock_ra[0:8] of the last page of the bit stream, the heap address lock_ba[0:1], the lock flag lock and 4’d0 in sequence.

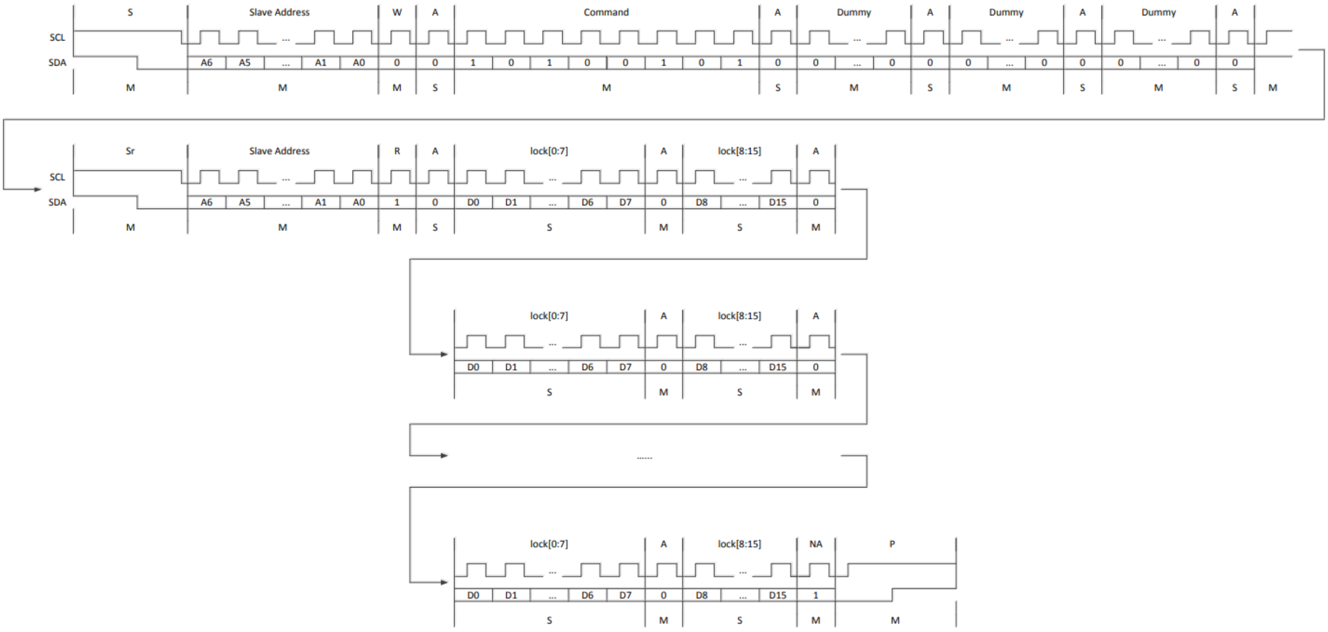
For 1K devices, lock_ba[1:0] is a fixed value of 2’b11.

For 2K devices, lock_ba[1:0] is the reserved heap address.

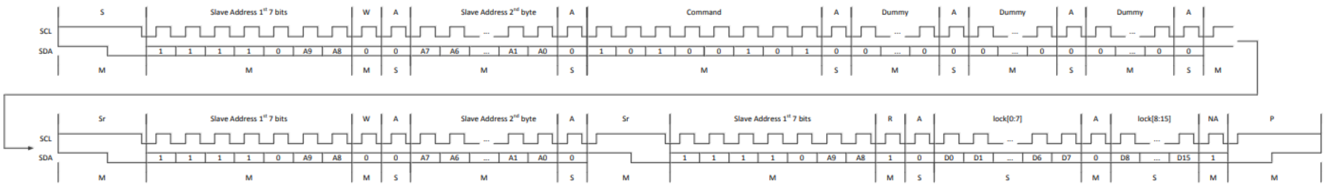
7 bit addressing



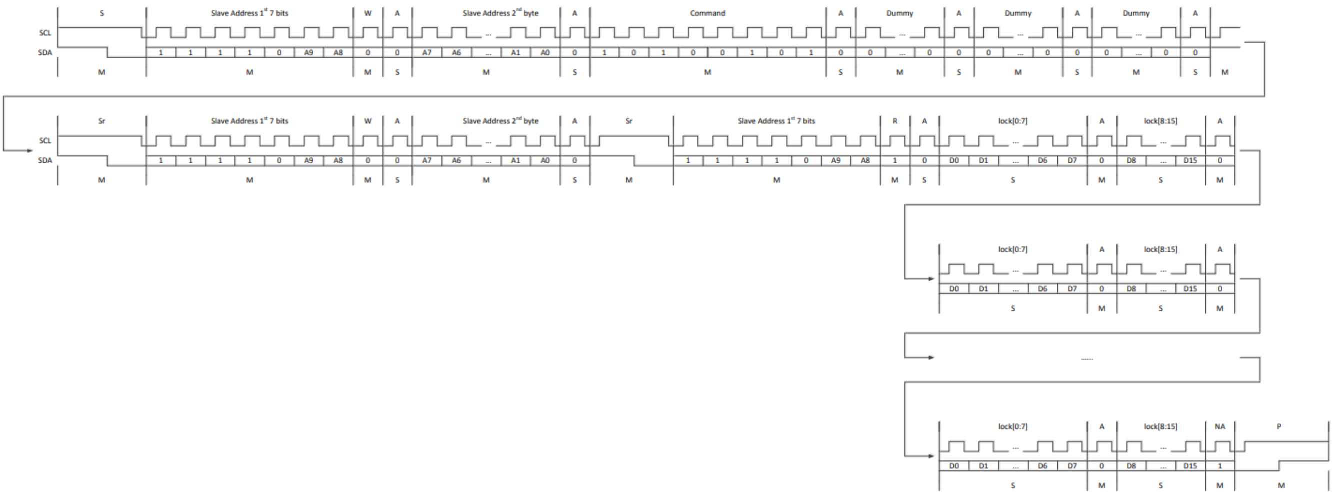
The lock information of the embedded Flash can be continuously read back.



10 bit addressing



The lock information of the embedded Flash can be continuously read back.

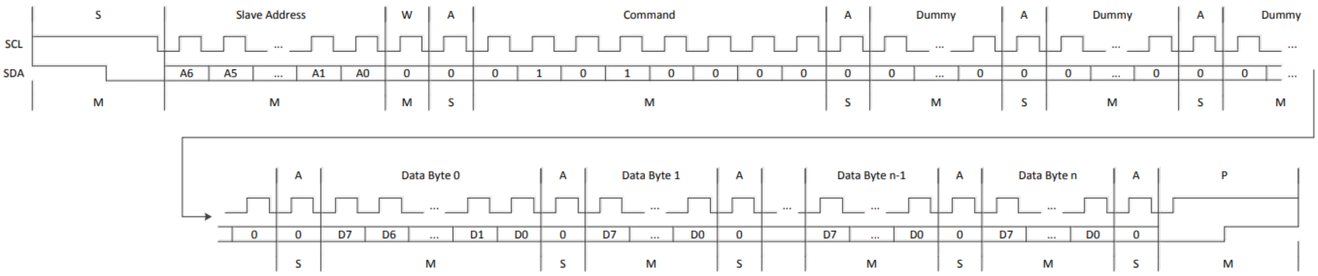


CFG

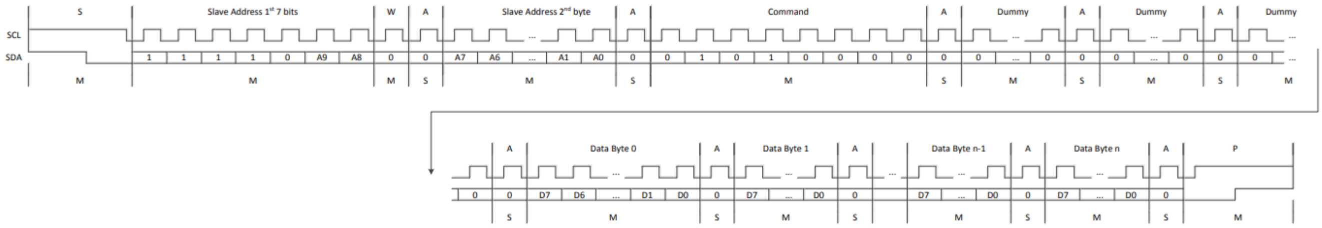
Configuration directives.

It is used to load the bit stream and configure the CPLD device. Write the high-order bit of each byte first.

7 bit addressing



10 bit addressing

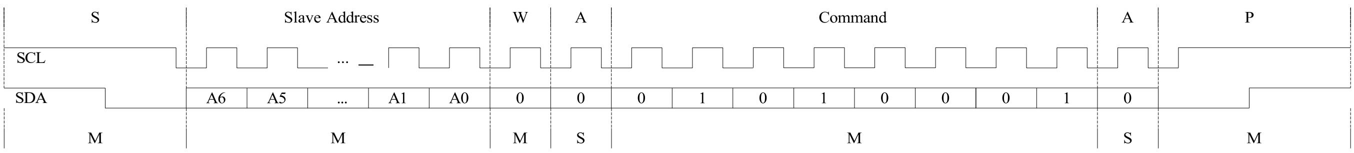


WREN

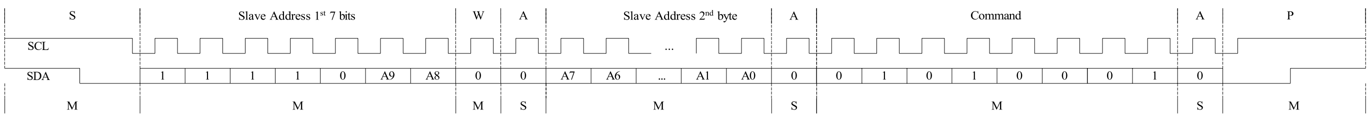
Write enable command.

Used before configuration directives and various programming directives.

7 bit addressing



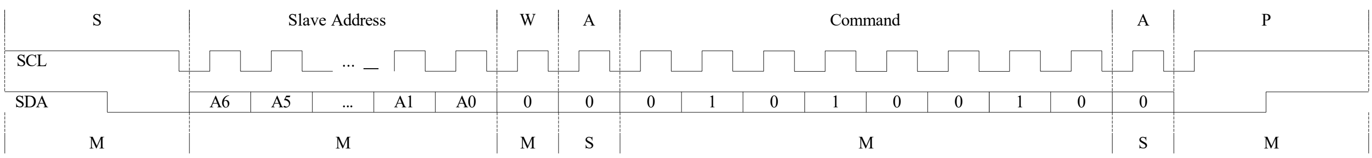
10 bit addressing



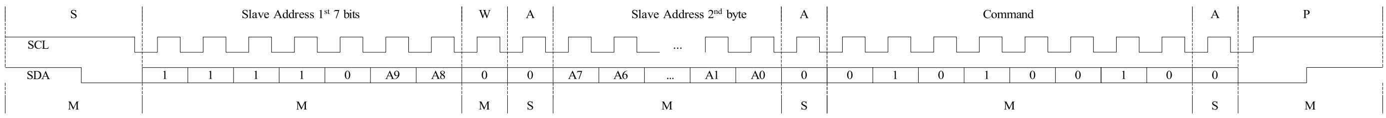
WRDIS

Write disable instruction.

7 bit addressing



10 bit addressing

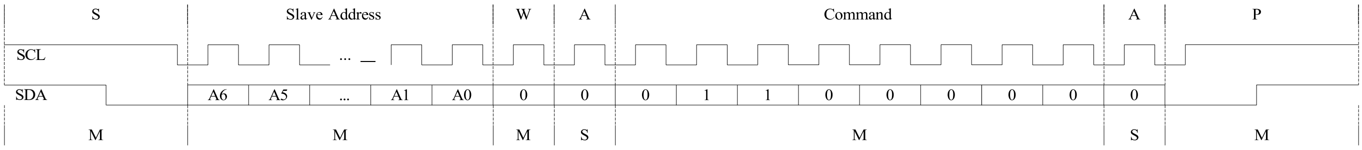


RESET

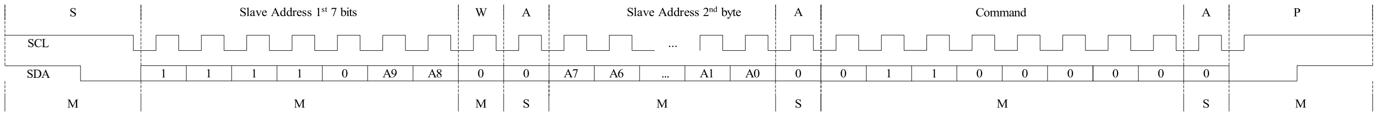
Reset command.

Used to reset parts of the configuration control system other than JTAG.

7 bit addressing



10 bit addressing

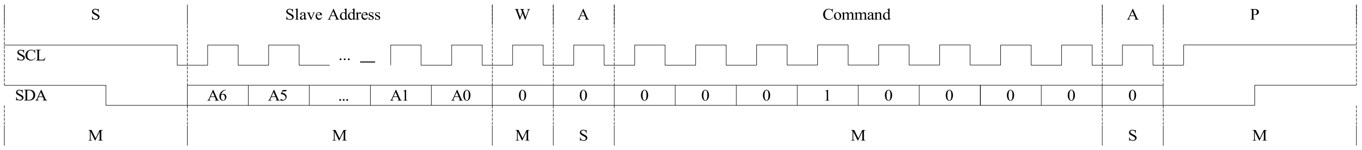


ERASE

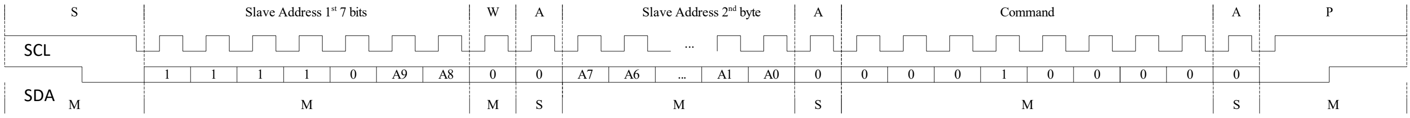
Erase command.

Used to erase embedded Flash all regular pages and clear embedded Flash Lock sign.

7 bit addressing



10 bit addressing

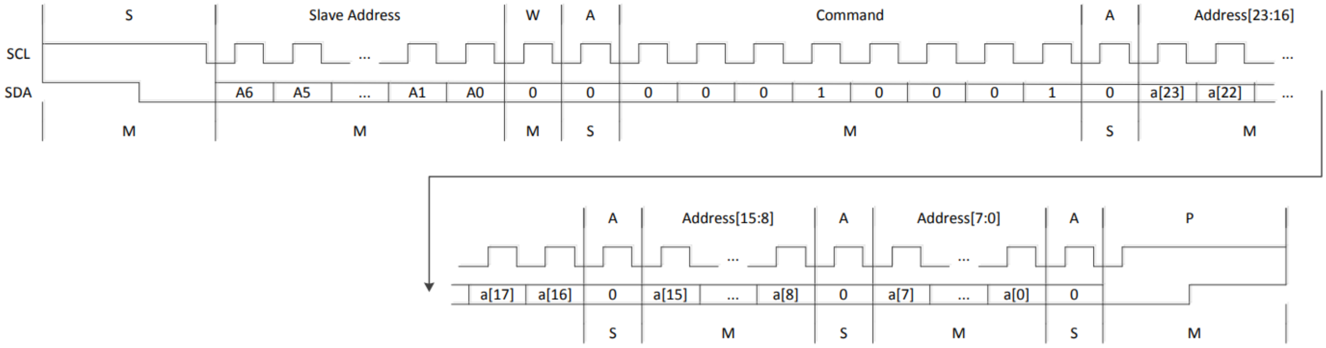


ERASE_PAGE

Page erase command.

Used to erase embedded Flash normal page.

7 bit addressing

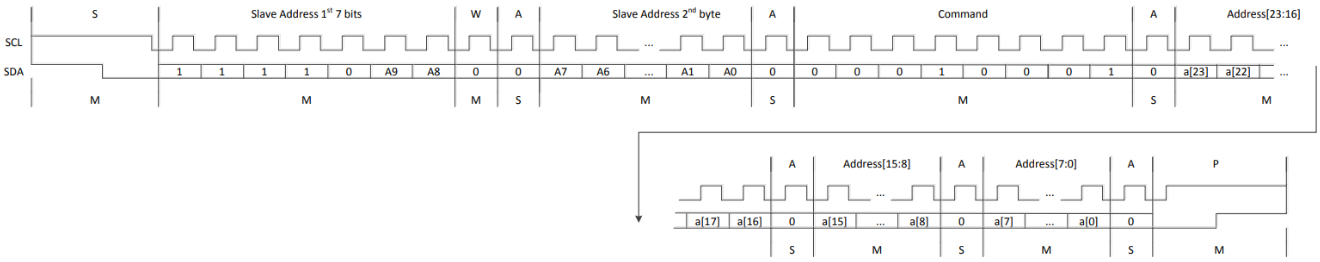


ADDRESS	NAME	DESCRIPTION
[23:19]	reserve	Need to be set to 5'b00000
[18:17]	Ba[1:0]	Heap address
[16:8]	Ra[8:0]	Page address
[7:0]	reserve	Needs to be set to 8'b00000000

for 1K devices, Ba[1:0] are reserved bits.

for 2K device, Ba[1:0] is the reserved heap address and needs to be set to 2'b11 .

10 bit addressing



ADDRESS	NAME	DESCRIPTION
[23:19]	reserve	Need to be set to 5'b00000
[18:17]	Ba[1:0]	heap address
[16:8]	Ra[8:0]	page address
[7:0]	reserve	Needs to be set to 8'b00000000

for 1K devices, Ba[1:0] are reserved bits.

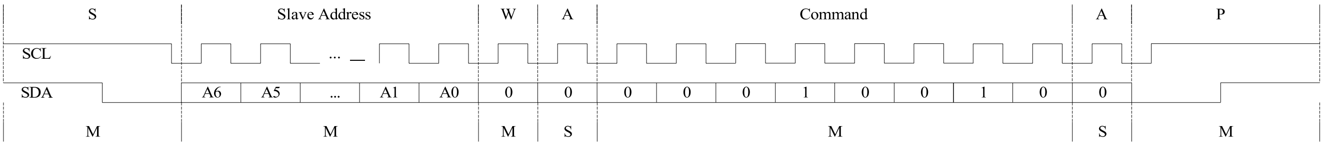
for 2K device, Ba[1:0] is the reserved heap address and needs to be set to 2'b11 .

ERASE_CTL

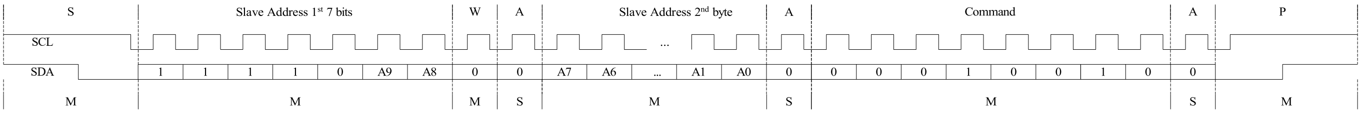
Controls the erase command.

Feature control page for erasing embedded Flash .

7 bit addressing



10 bit addressing

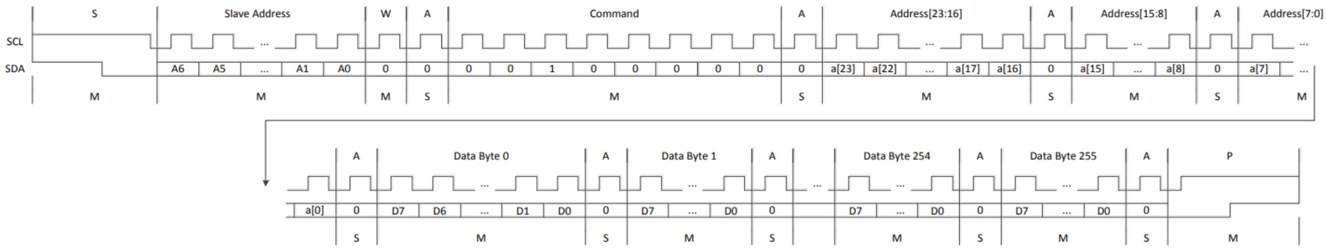


PROGRAM

Programming instructions.

Used to program the general pages of the embedded Flash.

7 bit addressing

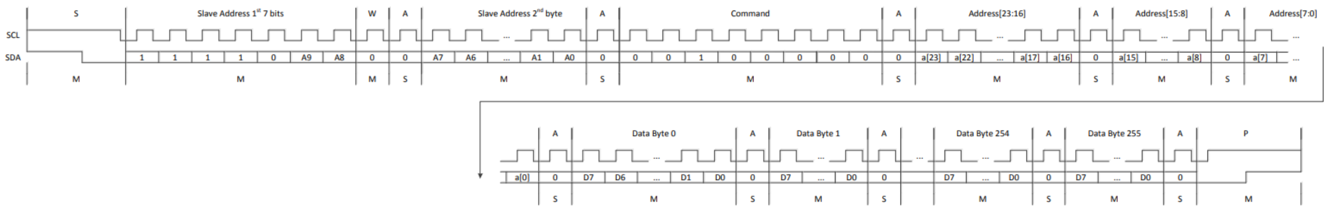


ADDRESS	NAME	DESCRIPTION
[23:19]	reserve	Need to be set to 5'b00000
[18:17]	Ba[1:0]	heap address
[16:8]	Ra[8:0]	page address
[7:0]	reserve	Needs to be set to 8'b00000000

for 1K devices, Ba[1:0] are reserved bits.

for 2K device, Ba[1:0] is the reserved heap address and needs to be set to 2'b11 .

10 bit addressing



ADDRESS	NAME	DESCRIPTION
[23:19]	reserve	Need to be set to 5'b00000

[18:17]	Ba[1:0]	heap address
[16:8]	Ra[8:0]	page address
[7:0]	reserve	Needs to be set to 8'b00000000

for 1K devices, Ba[1:0] are reserved bits.

for 2K device, Ba[1:0] is the reserved heap address and needs to be set to 2'b11 .

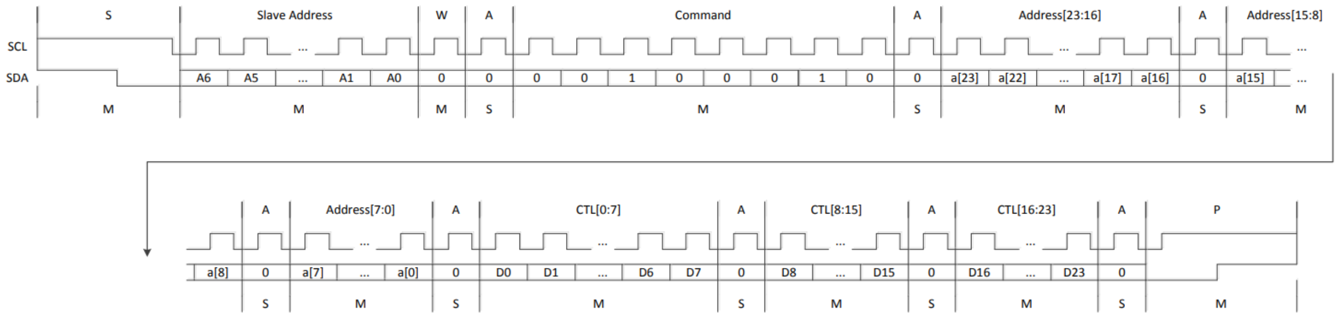
PROGRAM_CTL

Control programming instructions.

Used to program feature control bits.

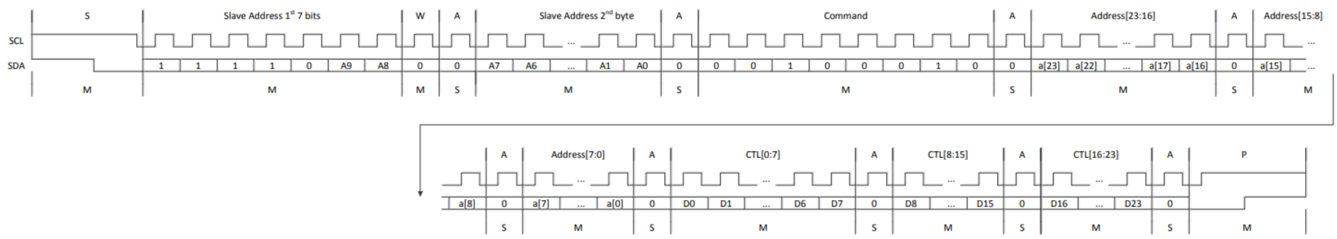
2K devices

7 bit addressing



ADDRESS	NAME	DESCRIPTION
[23:0]	reserve	Need to be set to 24'd0

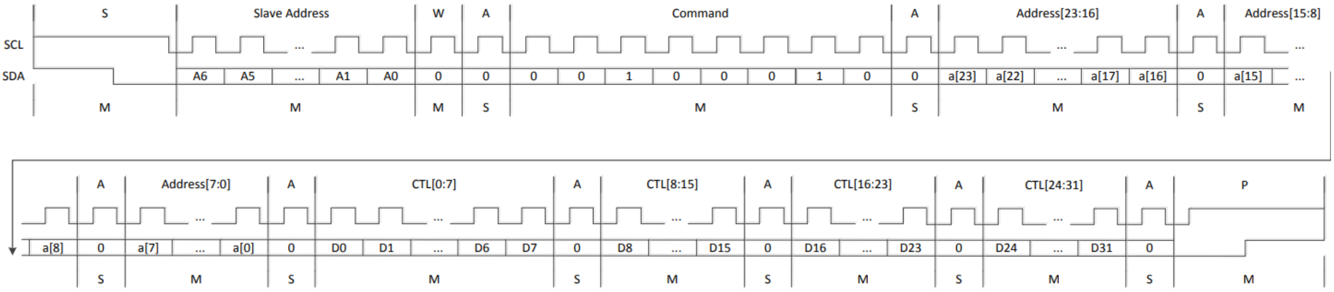
10 bit addressing



ADDRESS	NAME	DESCRIPTION
[23:0]	reserve	Need to be set to 24'd0

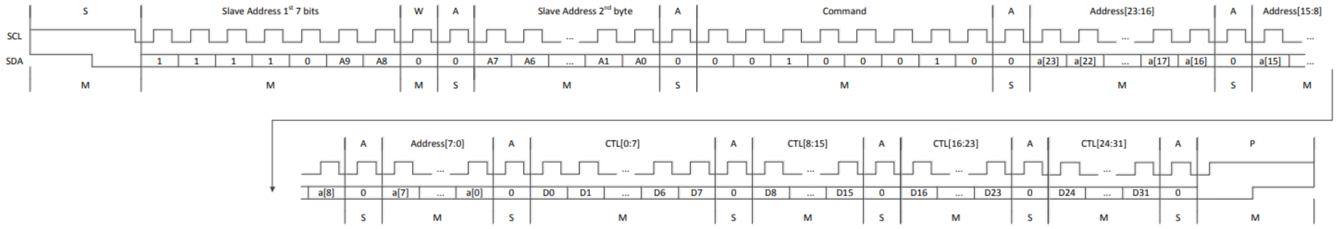
1K/4K/7K devices

7 bit addressing



ADDRESS	NAME	DESCRIPTION
[23:0]	reserve	Need to be set to 24'd0

10 bit addressing



ADDRESS	NAME	DESCRIPTION
[23:0]	reserve	Need to be set to 24'd0

READ

Read command.

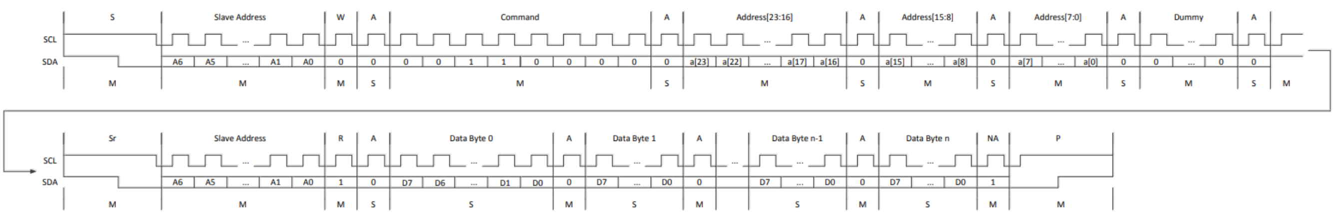
Used to read embedded Flash normal page.

ADDRESS	NAME	DESCRIPTION
[23:19]	reserve	Need to be set to 5'b00000
[18:17]	Ba[1:0]	heap address
[16:8]	Ra[8:0]	page address
[7:6]	reserve	Need to be set to 2'b00
[5:0]	Ca[5:0]	32 Bit data page offset address

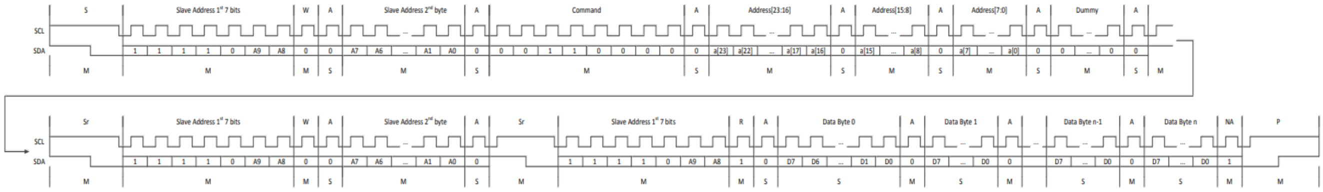
for 1K devices, Ba[1:0] are reserved bits.

for 2K device, Ba[1:0] is the reserved heap address and needs to be set to 2'b11 .

7 bit addressing



10 bit addressing



ADDRESS	NAME	DESCRIPTION
[23:19]	reserve	Need to be set to 5'b00000
[18:17]	Ba[1:0]	heap address
[16:8]	Ra[8:0]	page address
[7:6]	reserve	Need to be set to 2'b00
[5:0]	Ca[5:0]	32 Bit data page offset address

for 1K devices, Ba[1:0] are reserved bits.

for 2K device, Ba[1:0] is the reserved heap address and needs to be set to 2'b11 .

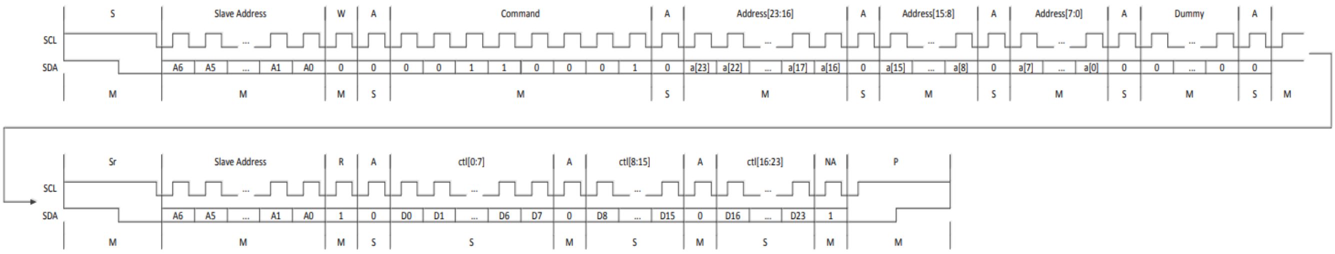
READ_CTL

Control read command.

Used to read embedded Flash feature control bits.

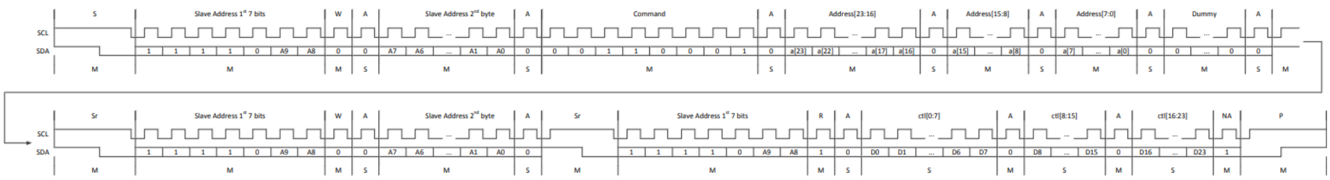
2K devices

7 bit addressing



ADDRESS	NAME	DESCRIPTION
[23:0]	reserve	Need to be set to 24'd0

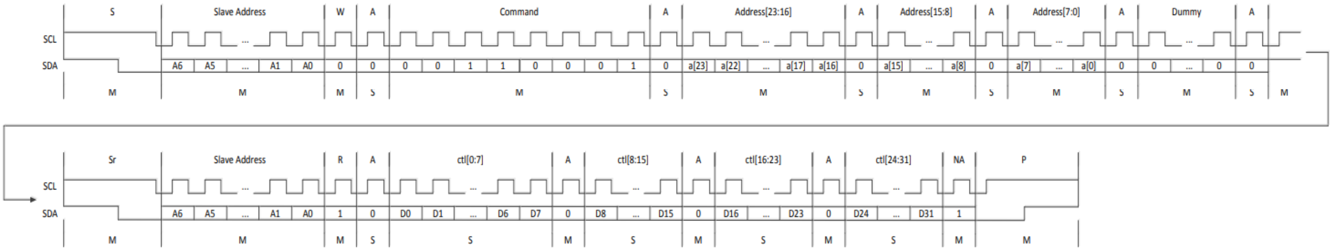
10 bit addressing



ADDRESS	NAME	DESCRIPTION
[23:0]	reserve	Need to be set to 24'd0

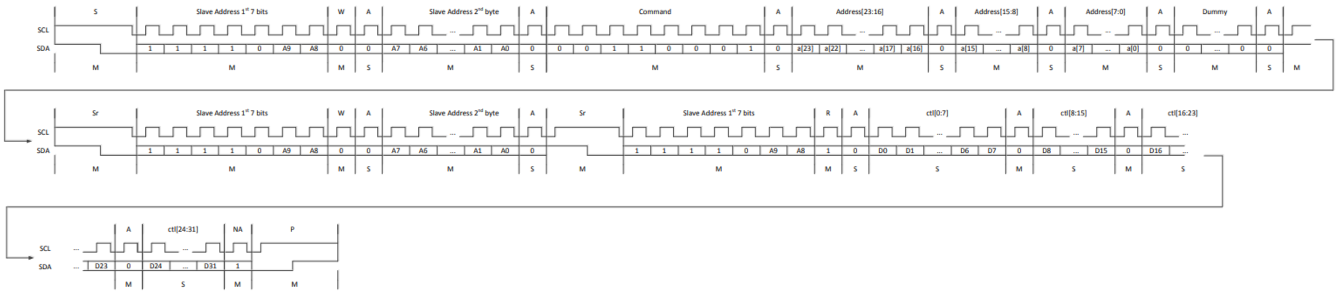
1K/4K/7K Devices

7 bit addressing



ADDRESS	NAME	DESCRIPTION
[23:0]	reserve	Need to be set to 24'd0

10 bit addressing



ADDRESS	NAME	DESCRIPTION
[23:0]	reserve	Need to be set to 24'd0

PROGRAM_LOCK

Lock instruction.

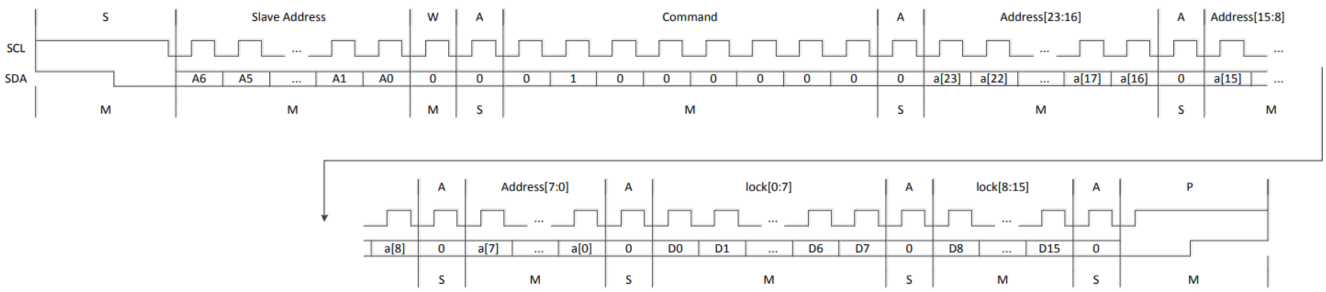
Used to inhibit programming and reading operations on the bit stream.

The data is sequentially shifted into the page address lock_ra[0:8] of the last page of the bitstream, the heap address lock_ba[0:1], the lock flag lock and 4’d0.

For 1K devices, lock_ba[1:0] are reserved bits.

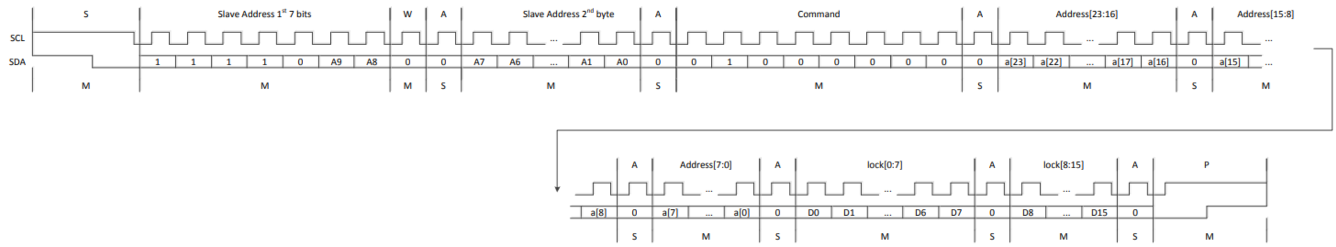
For 2K devices, lock_ba[1:0] is the reserved heap address and needs to be set to 2’b11.

7 bit addressing



ADDRESS	NAME	DESCRIPTION
[23:0]	reserve	Need to be set to 24'd0

10 bit addressing



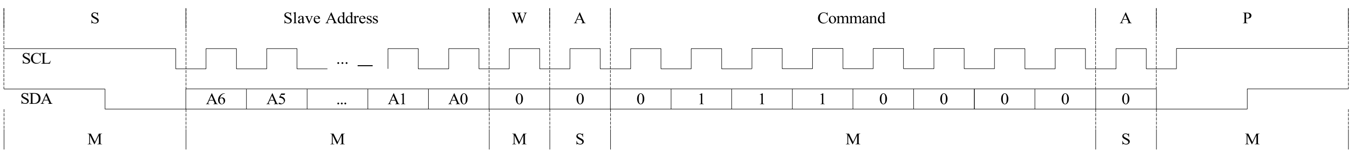
ADDRESS	NAME	DESCRIPTION
[23:0]	reserve	Need to be set to 24'd0

EFlash_SLEEP

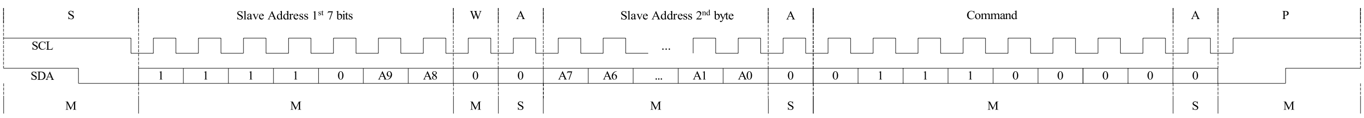
Embedded Flash sleep command.

Used to hibernate embedded Flash.

7 bit addressing



10 bit addressing

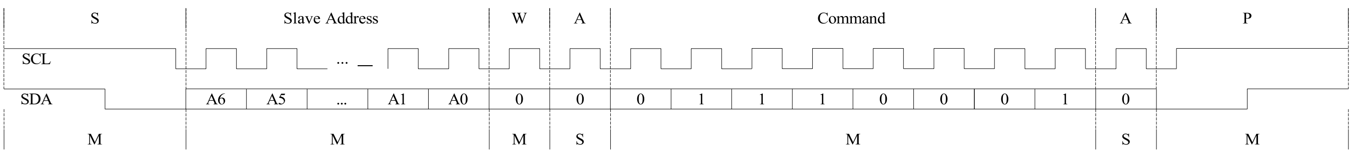


EFlash_WAKEUP

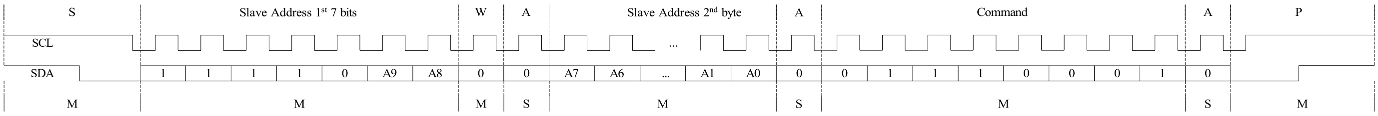
Embedded Flash wake-up command.

Used to wake up the embedded Flash.

7 bit addressing



10 bit addressing



Appendix 6

Internal slave APB interface register introduction.

IDCODE register 0

BIT	NAME	DESCRIPTION
[7:0]	idcode[7:0]	IDCODE[7:0]

IDCODE register 1

BIT	NAME	DESCRIPTION
[7:0]	idcode[15:8]	IDCODE[15:8]

IDCODE register 2

BIT	NAME	DESCRIPTION
[7:0]	idcode[23:16]	IDCODE[23:16]

IDCODE register 3

BIT	NAME	DESCRIPTION
[7:0]	idcode[31:24]	IDCODE[31:24]

USERCODE register 0

BIT	NAME	DESCRIPTION
[7:0]	usercode[7:0]	USERCODE[7:0]

USERCODE register 1

BIT	NAME	DESCRIPTION
[7:0]	usercode[15:8]	USERCODE[15:8]

USERCODE register 2

BIT	NAME	DESCRIPTION
[7:0]	usercode[23:16]	USERCODE[23:16]

USERCODE Register 3

BIT	NAME	DESCRIPTION
[7:0]	usercode[31:24]	USERCODE[31:24]

UID register 0

BIT	NAME	DESCRIPTION
[7:0]	uid[7:0]	UID[7:0]

UID register 1

BIT	NAME	DESCRIPTION
[7:0]	uid[15:8]	UID[15:8]

UID register 2

BIT	NAME	DESCRIPTION
[7:0]	uid[23:16]	UIDs [23:16]

UID Register 3

BIT	NAME	DESCRIPTION
[7:0]	uid[31:24]	UIDs [31:24]

UID Register 4

BIT	NAME	DESCRIPTION
[7:0]	uid[39:32]	UIDs [39:32]

UID Register 5

BIT	NAME	DESCRIPTION
[7:0]	uid[47:40]	UIDs [47:40]

UID Register 6

BIT	NAME	DESCRIPTION
[7:0]	uid[55:48]	UIDs [55:48]

UID Register 7

BIT	NAME	DESCRIPTION
[7:0]	uid[63:56]	UIDs [63:56]

Configuration Control System Status Register 0

BIT	NAME	DESCRIPTION
[7:0]	status[7:0]	Configure the status register [7:0] of the control system

Configuration Control System Status Register 1

BIT	NAME	DESCRIPTION
[7:0]	status[15:8]	Configure the status registers of the control system [15:8]

Configuration Control System Status Register 2

BIT	NAME	DESCRIPTION
[7:0]	status[23:16]	Configuring the status register of the control system [23:16]

Configuration Control System Status Register 3

BIT	NAME	DESCRIPTION
[7:0]	status[31:24]	Configuring the status register of the control system [31:24]

Interrupt Control Register

BIT	NAME	DESCRIPTION
[7:5]	reserve	
[4]	irq_rrdy	Read Ready Interrupt Enable 0 : disabled 1 : enable
[3]	irq_wrdy	Write Ready Interrupt Enable 0 : disabled 1 : enable
[2]	irq_idle	Embedded Flash Idle interrupt enable 0 : disabled 1 : enable
[1]	irq_si2c_proc	Interrupt enable from I^2C interface operation 0 : disabled 1 : enable
[0]	irq_sspi_proc	Interrupt enable from SPI interface operation 0 : disabled 1 : enable

Command register

INSTRUCTION	DESCRIPTION	OPCODE
NOP	no Operation	FF
WREN	Write enable	51
WRDIS	Write Disable	52
RESET	Reset CPLD device	60
ERASE	Erase Bulk	10
ERASE_PAGE	Erase Page	11
ERASE_CTL	Erase Features control bit	12
PROGRAM	program page	20
PROGRAM_CTL	program Features Control Bit	21
READ	Read	30
READ_CTL	Read Features control page	31
PROGRAM_LOCK	lock Embedded Flash	40
EFlash_SLEEP	Embedded Flash sleep	70
EFlash_WAKEUP	Embedded Flash wake up up	71

address register 0

BIT	NAME	DESCRIPTION
[7:6]	reserve	
[5:0]	Ca[5:0]	32 Bit data page offset address

address register 1

BIT	NAME	DESCRIPTION
[7:0]	Ra[7:0]	page address [7:0]

address register 2

2K

BIT	NAME	DESCRIPTION
[7:3]	reserve	
[2:1]	Ba[1:0]	heap address
[0]	Ra[8]	page address [8]

For 1K devices, Ba[1:0] are reserved bits.

For 2K devices, Ba[1:0] is the reserved heap address and needs to be set to 2'b11.

Data send register

Used to send embedded Flash programming data.

Data Receive Register

Used to receive embedded Flash read data.

status register

BIT	NAME	DESCRIPTION
[7:5]	reserve	
[4]	rrdy	read status indication 0 : not ready to read data 1 : Ready to read data, can read data
[3]	wrdy	programming status indication 0 : not ready to receive programming data 1 : Ready to receive programming data, can write programming data
[2]	busy	Embedded Flash busy indicator 0 : idle 1 : busy

[1]	si2c_proc	Slave I^2C Interface Status 0: idle 1: working
[0]	sspi_proc	Slave SPI interface status 0: idle 1: working

Interrupt Status Register

BIT	NAME	DESCRIPTION
[7:5]	reserve	
[4]	irq_rrdy	read ready interrupt, write 1 to clear 0 : not ready to read data 1 : Ready to read data, can read data
[3]	irq_wrdy	Write ready interrupt, write 1 to clear 0: Not ready to receive programming data 1: Ready to receive programming data, can write programming data
[2]	irq_idle	Embedded Flash Idle interrupt, write 1 to clear
[1]	irq_si2c_proc	Interrupt from I^2C interface, write 1 to clear
[0]	irq_sspi_proc	Interrupt from SPI interface, write 1 to clear

Interrupt register

BIT	NAME	DESCRIPTION
[7:5]	reserve	
[4]	irq_timer	Timer function block interrupt
[3]	irq_si2c1	I^2C Function block 1 to interrupt
[2]	irq_si2c0	I^2C function block 0 to interrupt
[1]	irq_spi	SPI function block interrupt
[0]	irq_ccs	Embedded Flash controller interrupt

Lock Page Address 0 register

BIT	NAME	DESCRIPTION
[7:0]	lock_ra[7:0]	Bit stream last page address low 8 bit

Lock Page Address 1 register

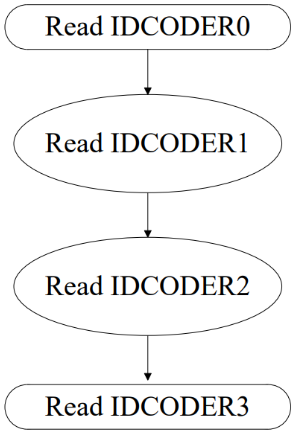
BIT	NAME	DESCRIPTION
[7:4]	reserve	
[3]	lock	lock flag
[2:0]	lock_ra[10:8]	bit stream tail page address high 3 bit

Appendix 7

Internal from APB operational process flow.

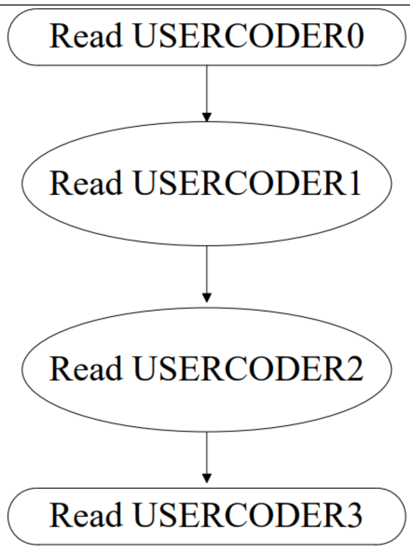
read IDCODE

read IDCODE register 0
read IDCODE register 1
read IDCODE register 2
read IDCODE Register 3



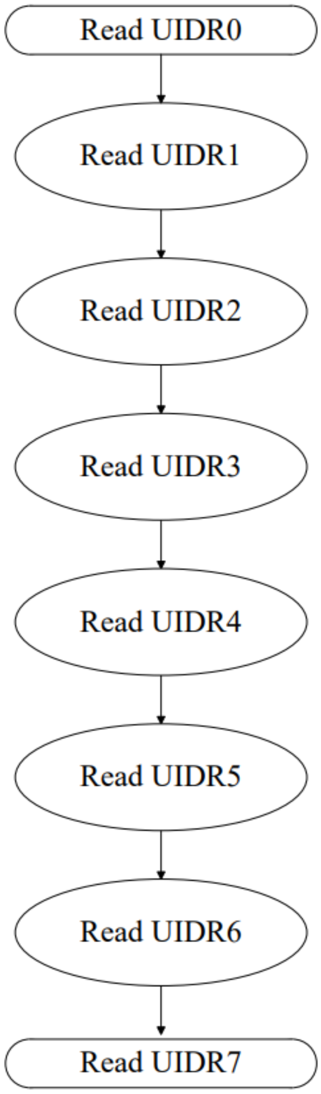
read USERCODE

read USERCODE register 0
read USERCODE register 1
read USERCODE register 2
read USERCODE Register 3



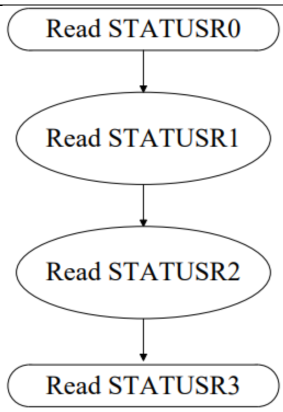
read UID

read UID register 0
read UID register 1
read UID register 2
read UID Register 3
read UID Register 4
read UID Register 5
read UID Register 6
read UID Register 7



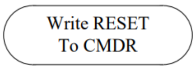
Read the status register of the configuration control system

Read Configuration Control System Status Register 0
Read Configuration Control System Status Register 1
Read Configuration Control System Status Register 2
Read Configuration Control System Status Register 3



reset

Write RESET command to command register

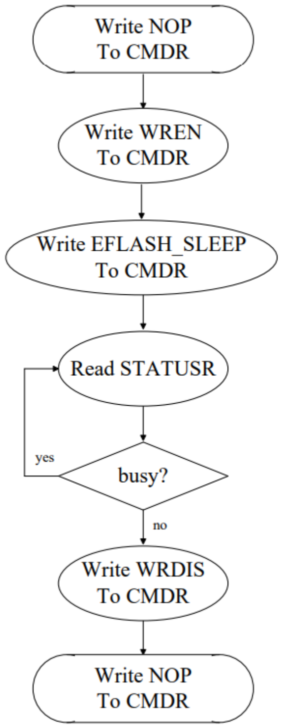


Hibernate Embedded Flash

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in an idle state (the busy bit of the status register is 0).

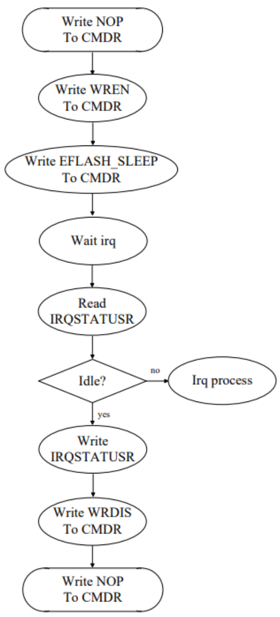
peer-to-peer mode

write NOP command to command register
write WREN command to command register
write EFlash_SLEEP command to command register
Read the status register continuously until busy is detected for low
write WRDIS command to command register
write NOP command to command register



interrupt mode

write NOP command to command register
write WREN command to command register
write EFlash_SLEEP command to command register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write WRDIS command to command register
write NOP command to command register

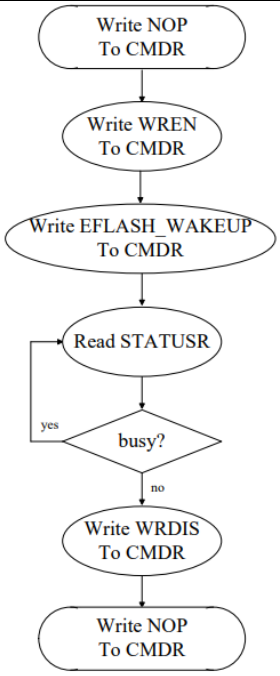


Wake up embedded Flash

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in an idle state (the busy bit of the status register is 0).

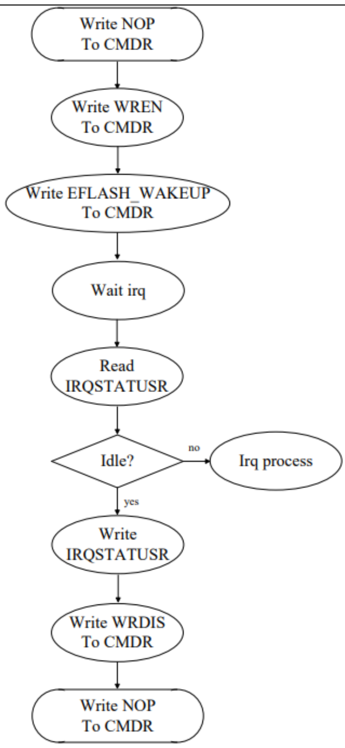
peer-to-peer mode

write NOP command to command register
write WREN command to command register
Write EFlash_WAKEUP command to command register
Read the status register continuously until busy is detected for low
write WRDIS command to command register
write NOP command to command register



interrupt mode

write NOP command to command register
write WREN command to command register
Write EFlash_WAKEUP command to command register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write WRDIS command to command register



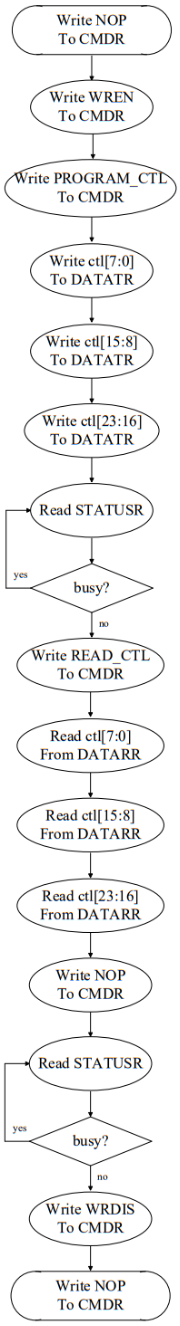
Programming Feature Control Bits

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in an idle state (the busy bit of the status register is 0).

peer-to-peer mode

write NOP command to command register
write WREN command to command register
write PROGRAM_CTL command to command register
Write feature control bits ctl[7:0] to data transmit register
Write feature control bits ctl[15:8] to data transmit register
Write feature control bits ctl[23:16] to data transmit register
Read the status register continuously until busy is detected for low
write READ_CTL command to command register
Read data receive register, read feature control bit ctl[7:0]
Read data receive register, read feature control bit ctl[15:8]
Read data receive register, read feature control bit ctl[23:16]
write NOP command to command register
Read the status register continuously until busy is detected for low

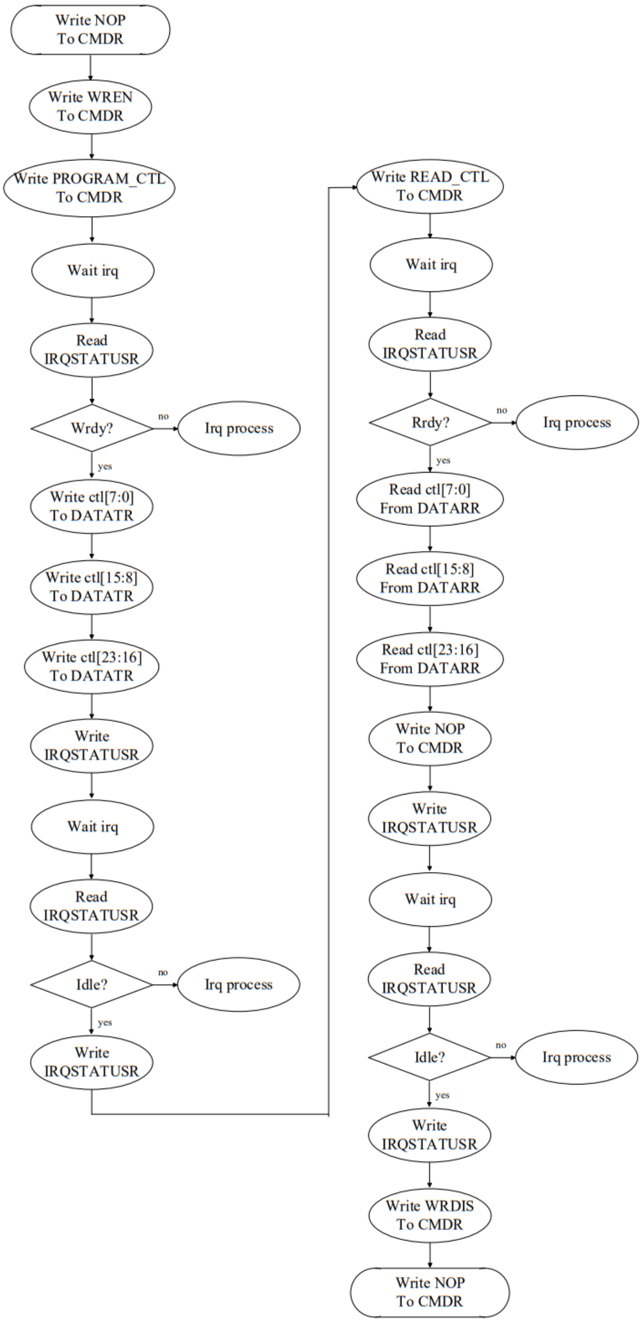
write WRDIS command to command register
write NOP command to command register



interrupt mode

write NOP command to command register
write WREN command to command register
write PROGRAM_CTL command to command register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash write ready interrupt
Write feature control bits ctl[7:0] to data transmit register
Write feature control bits ctl[15:8] to data transmit register
Write feature control bits ctl[23:16] to data transmit register
Write interrupt status register, clear embedded Flash write ready interrupt

After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write READ_CTL command to command register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash read ready interrupt
Read data receive register, read feature control bit ctl[7:0]
Read data receive register, read feature control bit ctl[15:8]
Read data receive register, read feature control bit ctl[23:16]
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write WRDIS command to command register
write NOP command to command register



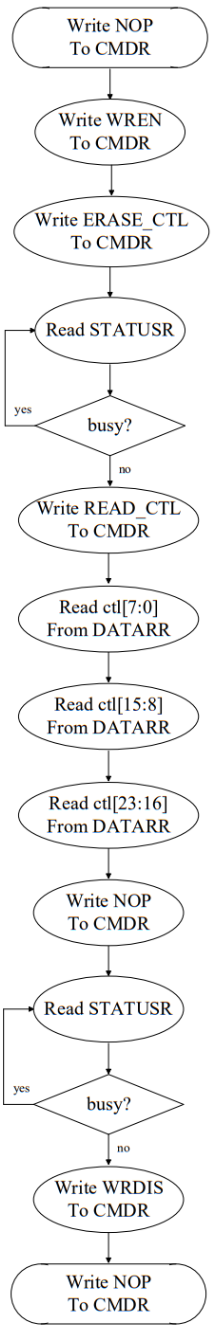
reset feature control bit

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in an idle state (the busy bit of the status register is 0).

peer-to-peer mode

write NOP command to command register
write WREN command to command register
Write ERASE_CTL command to command register
Read the status register continuously until busy is detected for low
write READ_CTL command to command register
Read data receive register, read feature control bit ctl[7:0]
Read data receive register, read feature control bit ctl[15:8]

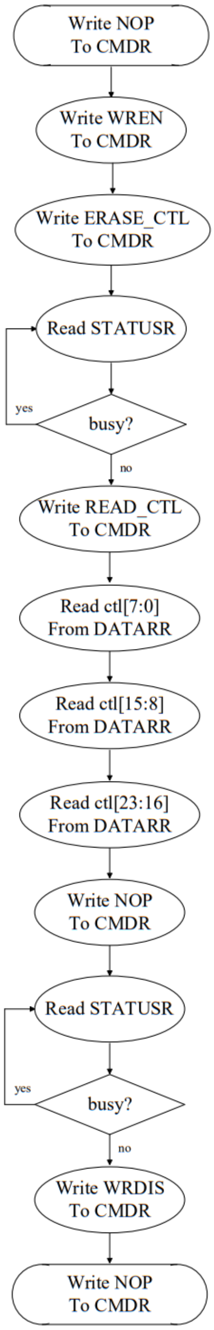
Read data receive register, read feature control bit ctl[23:16]
write NOP command to command register
Read the status register continuously until busy is detected for low
write WRDIS command to command register
write NOP command to command register



interrupt mode

write NOP command to command register
write WREN command to command register
Write ERASE_CTL command to command register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt

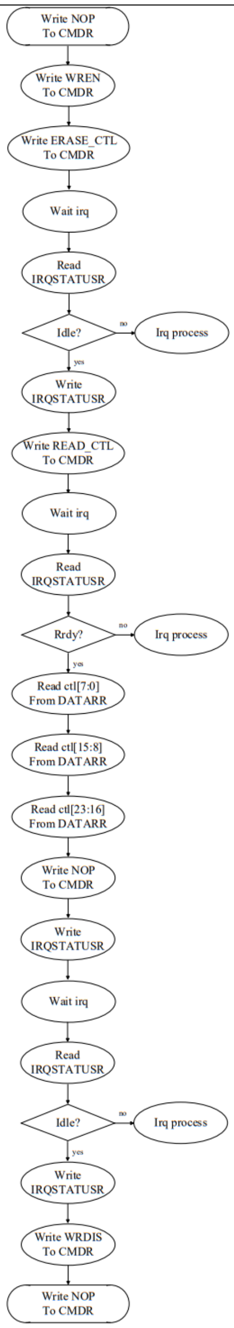
write READ_CTL command to command register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash read ready to interrupt
Read data receive register, read feature control bit ctl[7:0]
Read data receive register, read feature control bit ctl[15:8]
Read data receive register, read feature control bit ctl[23:16]
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write WRDIS command to command register
write NOP command to command register



interrupt mode

Write NOP command to command register
Write WREN command to command register
Write ERASE_CTL command to command register
After receiving an interrupt, read the interrupt status register to detect the embedded Flash idle interrupt
Write the interrupt status register to clear the embedded Flash idle interrupt
Write READ_CTL command to command register
After receiving an interrupt, read the interrupt status register to detect that the embedded Flash read is ready
to interrupt
Read data receive register, read feature control bits ctl[7:0]
Read data receive register, read feature control bits ctl[15:8]

Read data receive register, read feature control bits ctl[23:16]
Write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving an interrupt, read the interrupt status register to detect the embedded Flash idle interrupt
Write the interrupt status register to clear the embedded Flash idle interrupt
Write WRDIS command to command register
Write NOP command to command register

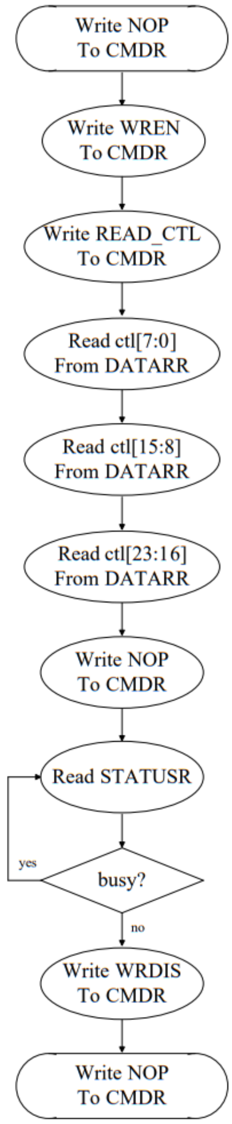


read feature control bits

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in an idle state (the busy bit of the status register is 0).

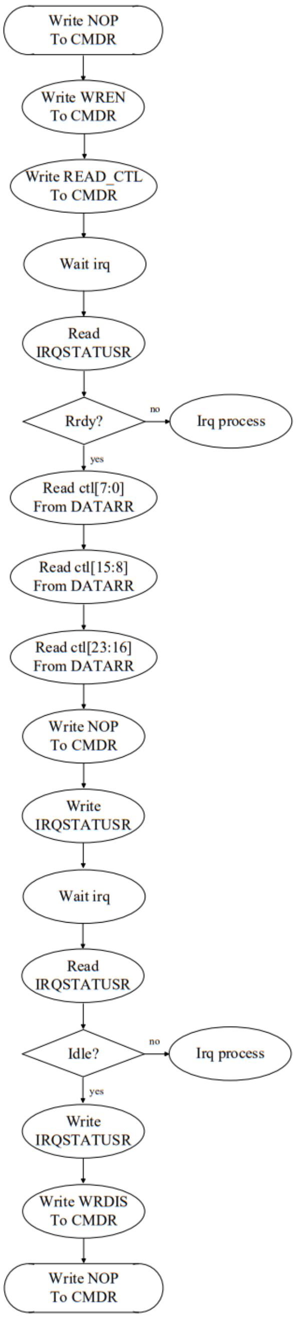
peer-to-peer mode

write NOP command to command register
write WREN command to command register
write READ_CTL command to command register
Read data receive register, read feature control bit ctl[7:0]
Read data receive register, read feature control bit ctl[15:8]
Read data receive register, read feature control bit ctl[23:16]
write NOP command to command register
Read the status register continuously until busy is detected for low
write WRDIS command to command register
write NOP command to command register



interrupt mode

write NOP command to command register
write WREN command to command register
write READ_CTL command to command register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash read ready interrupt
Read data receive register, read feature control bit ctl[7:0]
Read data receive register, read feature control bit ctl[15:8]
Read data receive register, read feature control bit ctl[23:16]
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write WRDIS command to command register
write NOP command to command register



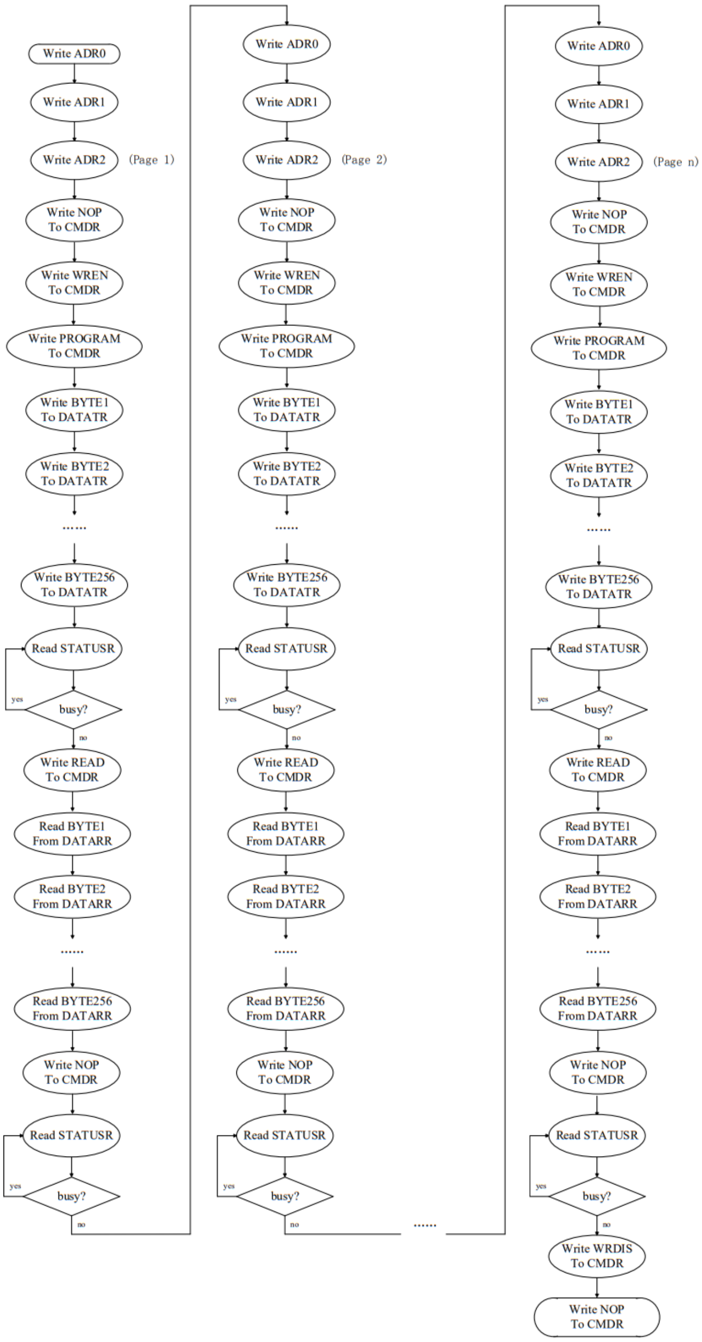
programming bit stream

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in an idle state (the busy bit of the status register is 0)

peer-to-peer mode

Write Address Register 0
Write Address Register 1
Write Address Register 2 , Write Bit Stream 1 page data address
write NOP command to command register
write WREN command to command register

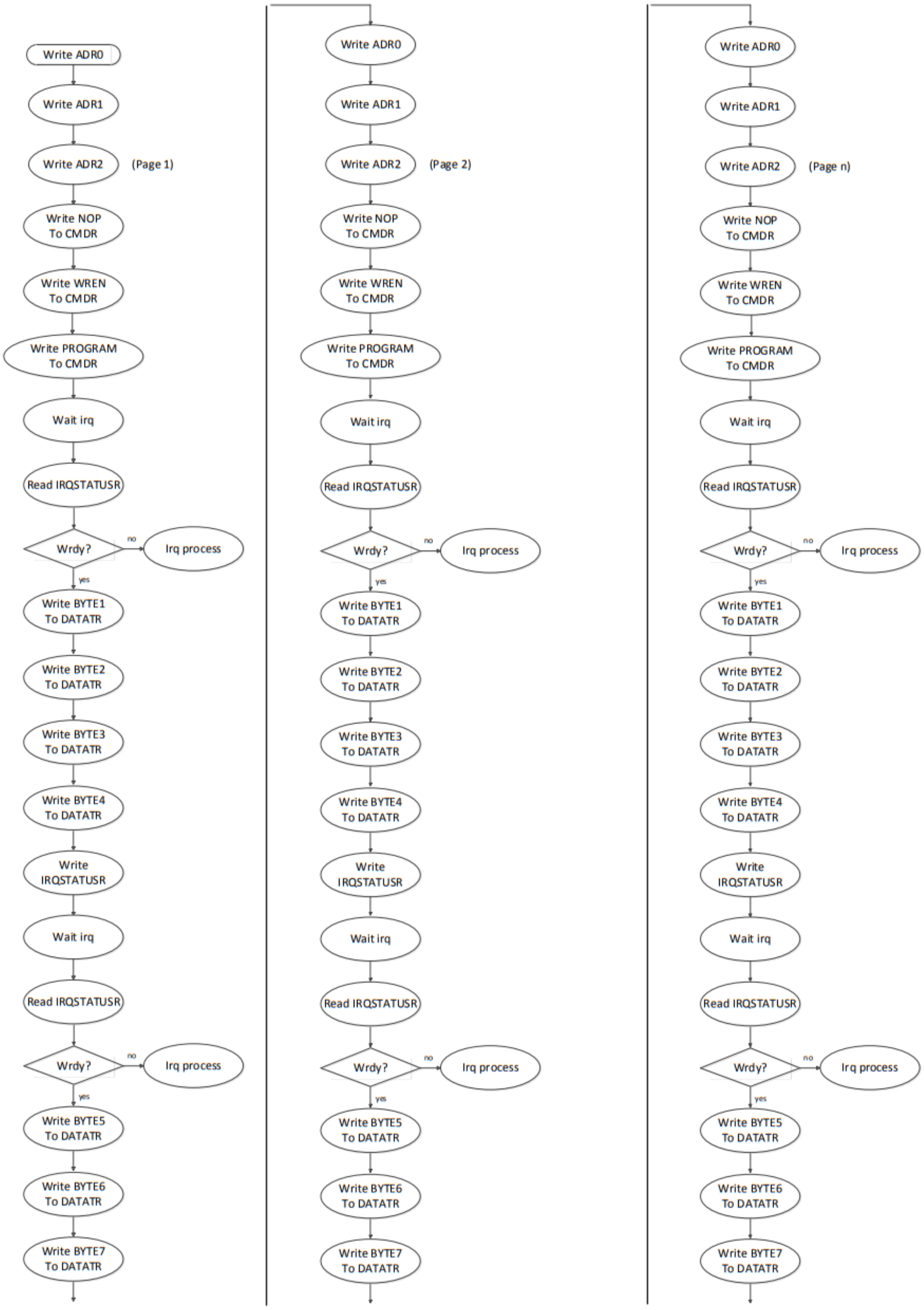
WRITE PROGRAM command to command register
write bitstream 1st Page Data to Data Transmit Register
Read the status register continuously until busy is detected for low
write read command to command register
Read data receive register, read bit stream 1 page data
write NOP command to command register
Read the status register continuously until busy is detected for low
Write Address Register 0
Write Address Register 1
Write Address Register 2 , Write Bit Stream 2 page data address
WRITE PROGRAM command to command register
write bitstream 2 Page Data to Data Transmit Register
Read the status register continuously until busy is detected for low
write read command to command register
Read data receive register, read bit stream 2 page data
write NOP command to command register
Read the status register continuously until busy is detected for low
...
Write Address Register 0
Write Address Register 1
Write address register 2 , write bit stream nth page data address
WRITE PROGRAM command to command register
write bitstream nth Page Data to Data Transmit Register
Read the status register continuously until busy is detected for low
write read command to command register
Read the data receiving register, read the nth bit stream page data
write NOP command to command register
Read the status register continuously until busy is detected for low
write WRDIS command to command register
write NOP command to command register

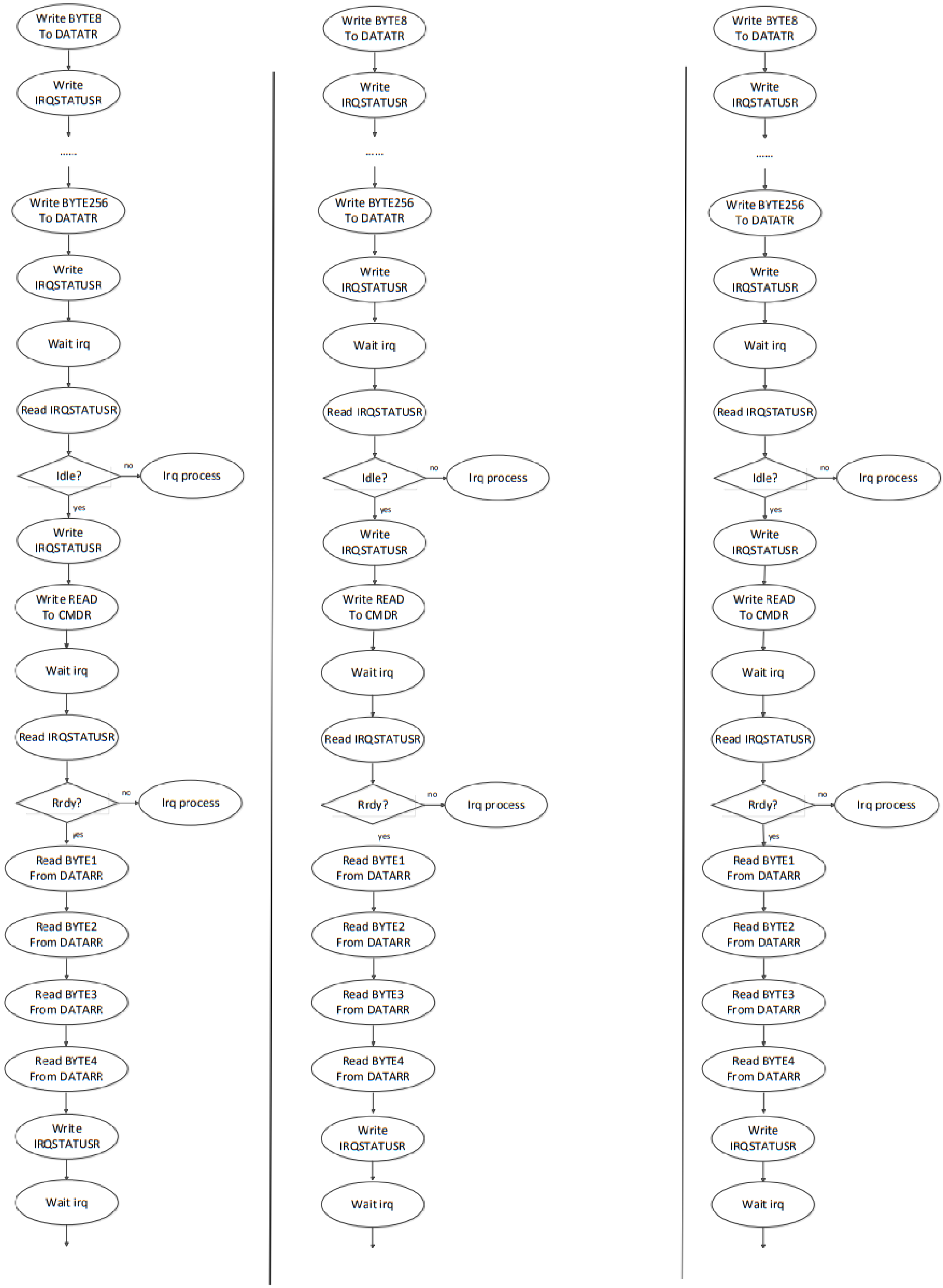

Write ADR0
Write ADR1
Write ADR2
(Page 1)
Write NOP To CMDR
Write WREN To CMDR
Write PROGRAM To CMDR
Write BYTE1 To DATATR
Write BYTE2 To DATATR
......
Write BYTE256 To DATATR
Read STATUSR
busy?
yes
no
Write READ To CMDR
Read BYTE1 From DATARR
Read BYTE2 From DATARR
......
Read BYTE256 From DATARR
Write NOP To CMDR
Read STATUSR
busy?
yes
no
Write ADR0
Write ADR1
Write ADR2
(Page 2)
Write NOP To CMDR
Write WREN To CMDR
Write PROGRAM To CMDR
Write BYTE1 To DATATR
Write BYTE2 To DATATR
......
Write BYTE256 To DATATR
Read STATUSR
busy?
yes
no
Write READ To CMDR
Read BYTE1 From DATARR
Read BYTE2 From DATARR
......
Read BYTE256 From DATARR
Write NOP To CMDR
Read STATUSR
busy?
yes
no
......
Write ADR0
Write ADR1
Write ADR2
(Page n)
Write NOP To CMDR
Write WREN To CMDR
Write PROGRAM To CMDR
Write BYTE1 To DATATR
Write BYTE2 To DATATR
......
Write BYTE256 To DATATR
Read STATUSR
busy?
yes
no
Write READ To CMDR
Read BYTE1 From DATARR
Read BYTE2 From DATARR
......
Read BYTE256 From DATARR
Write NOP To CMDR
Read STATUSR
busy?
yes
no
Write WRDIS To CMDR
Write NOP To CMDR

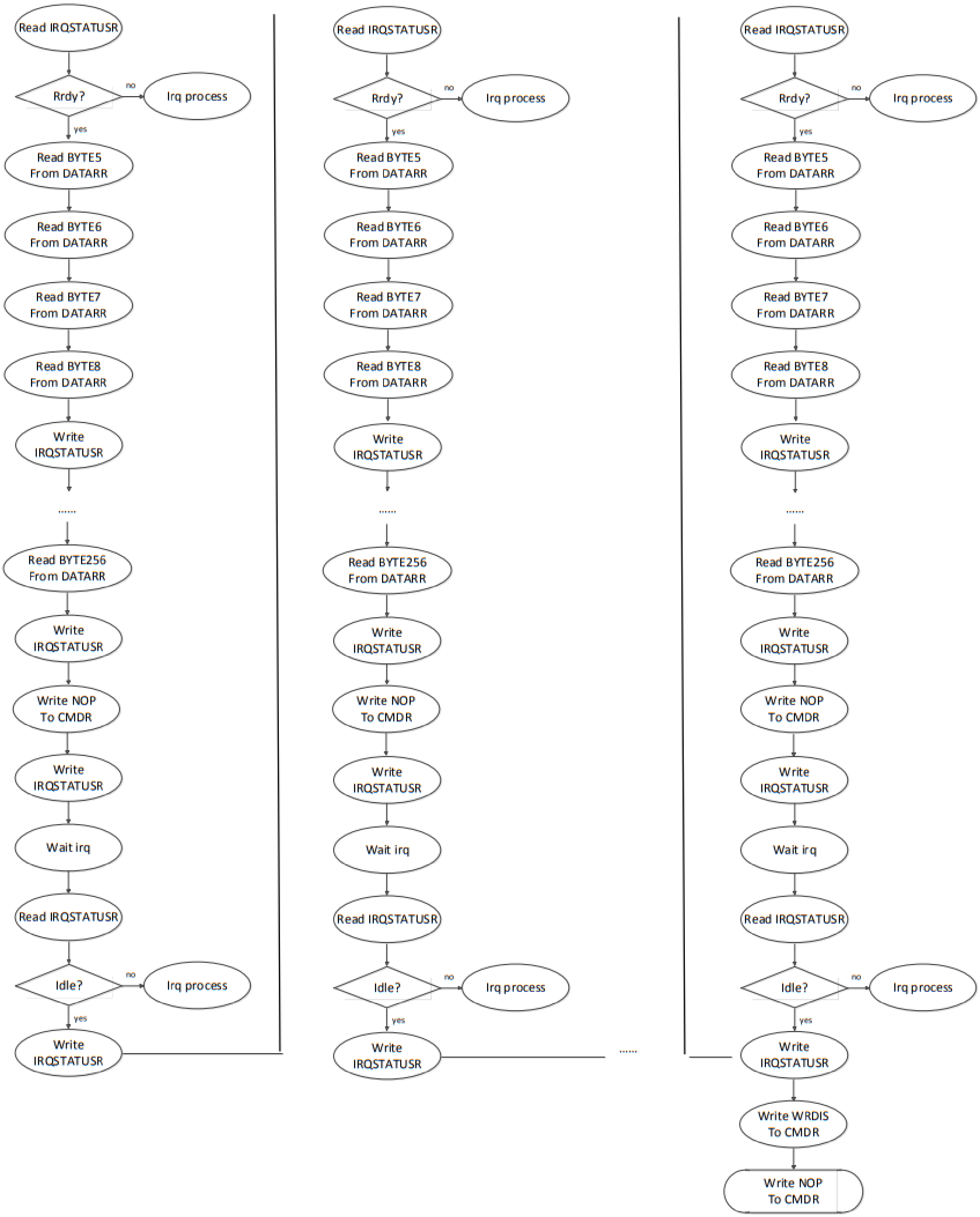
interrupt mode

Write Address Register 0
Write Address Register 1
Write Address Register 2 , Write Bit Stream 1 page data address
write NOP command to command register
write WREN command to command register
WRITE PROGRAM command to command register
write bitstream 1st Page data to data transmit register: After receiving the interrupt, read the interrupt status register and detect the embedded Flash Write ready interrupt writes the first byte to the data transmission register write 2nd bytes to data transmit register write 3rd bytes to data transmit register write number 4 bytes to data transmit register Write interrupt status register, clear embedded Flash write ready interrupt After receiving the interrupt, read the interrupt status register and detect the embedded Flash Write ready interrupt writes the 5th byte to the data transmit register write number 6 bytes to data transmit register write the 7th bytes to data transmit register write the 8th bytes to data transmit register Write interrupt status register, clear embedded Flash write ready interrupt … After receiving the interrupt, read the interrupt status register and detect the embedded Flash Write ready interrupt writes the 253rd byte to the data transmit register write the 254th bytes to data transmit register write the 255th bytes to data transmit register Write No. 256 bytes to data transmit register Write interrupt status register, clear embedded Flash write ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write read command to command register
Read data receive register, read bit stream 1 Page data: After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the first byte Read data receive register, read the 2nd bytes Read data receive register, read the 3rd bytes Read the data receive register, read the 4th bytes Write interrupt status register, clear embedded Flash read ready interrupt After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the 5th byte Read the data receive register, read the 6th bytes Read the data receive register, read the 7th bytes Read the data receive register, read the 8th bytes Write interrupt status register, clear embedded Flash read ready interrupt

... After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the 253rd byte Read data receive register, read No. 254 bytes Read the data receiving register, read the 255th bytes Read data receive register, read No. 256 bytes Write interrupt status register, clear embedded Flash read ready interrupt
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
Write Address Register 0
Write Address Register 1
Write Address Register 2 , Write Bit Stream 2 page data address
WRITE PROGRAM command to command register
write bitstream 2 Page Data to Data Transmit Register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write read command to command register
Read data receive register, read bit stream 2 page data
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
...
Write Address Register 0
Write Address Register 1
Write address register 2 , write bit stream nth page data address
WRITE PROGRAM command to command register
write bitstream nth Page Data to Data Transmit Register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write read command to command register
Read the data receiving register, read the nth bit stream page data
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write WRDIS command to command register
write NOP command to command register


Write ADR0
Write ADR1
Write ADR2
(Page 1)
Write NOP To CMDR
Write WREN To CMDR
Write PROGRAM To CMDR
Wait irq
Read IRQSTATUSR
Wrdy?
no
Irq process
yes
Write BYTE1 To DATATR
Write BYTE2 To DATATR
Write BYTE3 To DATATR
Write BYTE4 To DATATR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Wrdy?
no
Irq process
yes
Write BYTE5 To DATATR
Write BYTE6 To DATATR
Write BYTE7 To DATATR
(Page 2)
(Page n)


Write BYTE8 To DATATR
Write IRQSTATUSR
......
Write BYTE256 To DATATR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Idle?
no
Irq process
yes
Write IRQSTATUSR
Write READ To CMDR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE1 From DATARR
Read BYTE2 From DATARR
Read BYTE3 From DATARR
Read BYTE4 From DATARR
Write IRQSTATUSR
Wait irq
Write BYTE8 To DATATR
Write IRQSTATUSR
......
Write BYTE256 To DATATR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Idle?
no
Irq process
yes
Write IRQSTATUSR
Write READ To CMDR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE1 From DATARR
Read BYTE2 From DATARR
Read BYTE3 From DATARR
Read BYTE4 From DATARR
Write IRQSTATUSR
Wait irq
Write BYTE8 To DATATR
Write IRQSTATUSR
......
Write BYTE256 To DATATR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Idle?
no
Irq process
yes
Write IRQSTATUSR
Write READ To CMDR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE1 From DATARR
Read BYTE2 From DATARR
Read BYTE3 From DATARR
Read BYTE4 From DATARR
Write IRQSTATUSR
Wait irq



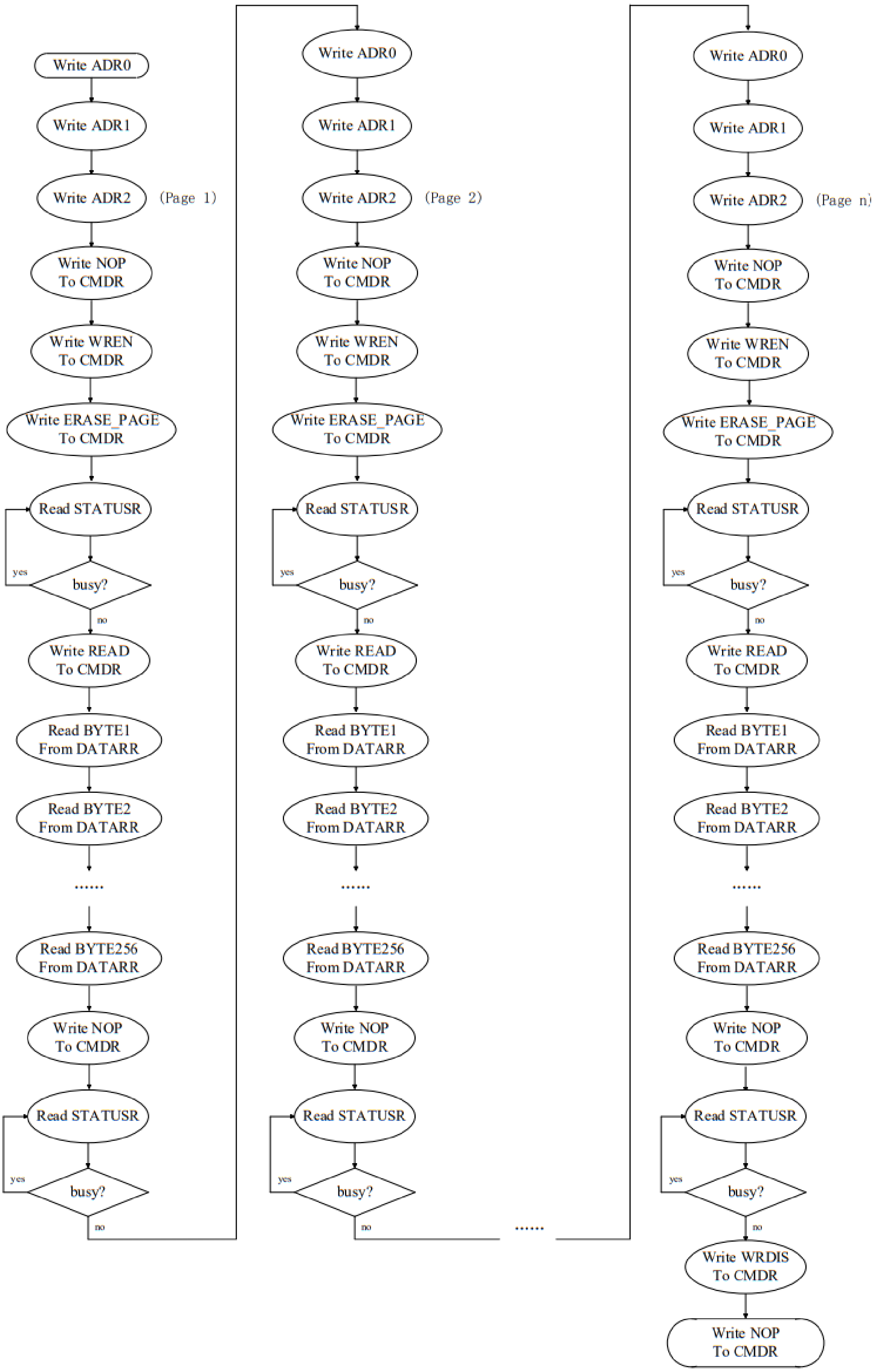
Erase bit stream

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in an idle state (the busy bit of the status register is 0).

peer-to-peer mode

Write Address Register 0

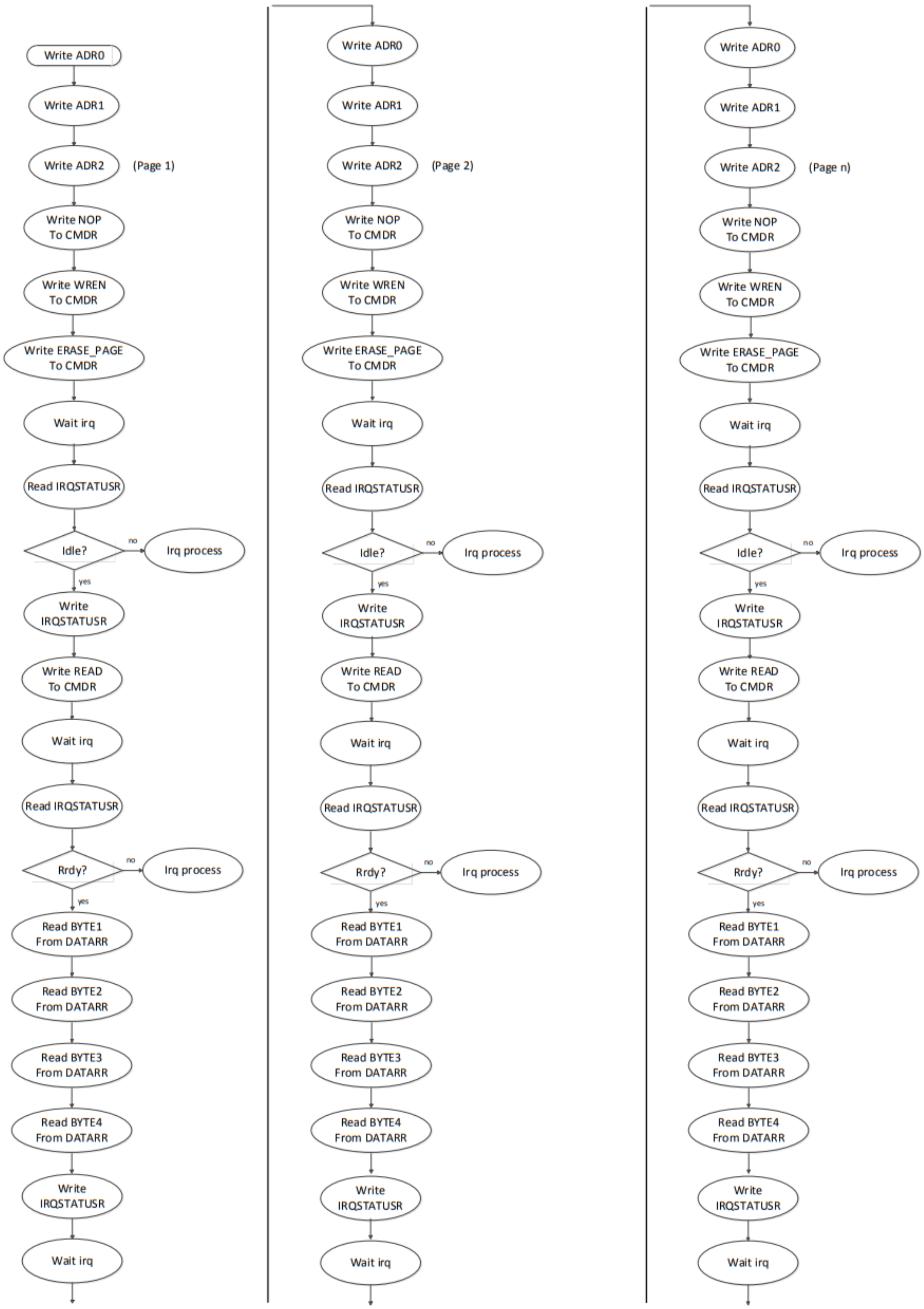
Write Address Register 1
Write Address Register 2 , Write Bit Stream 1 page data address
write NOP command to command register
write WREN command to command register
write ERASE_PAGE command to command register
Read the status register continuously until busy is detected for low
write read command to command register
Read data receive register, read bit stream 1 page data
write NOP command to command register
Read the status register continuously until busy is detected for low
Write Address Register 0
Write Address Register 1
Write Address Register 2 , Write Bit Stream 2 page data address
write NOP command to command register
write ERASE_PAGE command to command register
Read the status register continuously until busy is detected for low
write read command to command register
Read data receive register, read bit stream 2 page data
write NOP command to command register
Read the status register continuously until busy is detected for low
...
Write Address Register 0
Write Address Register 1
Write address register 2 , write bit stream nth page data address
write NOP command to command register
write ERASE_PAGE command to command register
Read the status register continuously until busy is detected for low
write read command to command register
Read the data receiving register, read the nth bit stream page data
write NOP command to command register
Read the status register continuously until busy is detected for low
write WRDIS command to command register
write NOP command to command register


Write ADR0
Write ADR1
Write ADR2
(Page 1)
Write NOP To CMDR
Write WREN To CMDR
Write ERASE_PAGE To CMDR
Read STATUSR
busy?
yes
no
Write READ To CMDR
Read BYTE1 From DATARR
Read BYTE2 From DATARR
......
Read BYTE256 From DATARR
Write NOP To CMDR
Read STATUSR
busy?
yes
no
(Page 2)
(Page n)
Write WRDIS To CMDR
Write NOP To CMDR

interrupt mode

Write Address Register 0
Write Address Register 1
Write Address Register 2 , Write Bit Stream 1 page data address
write NOP command to command register
write WREN command to command register
write ERASE_PAGE command to command register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write read command to command register
Read data receive register, read bit stream 1 Page data: After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the first byte Read data receive register, read the 2nd bytes Read data receive register, read the 3rd bytes Read the data receive register, read the 4th bytes Write interrupt status register, clear embedded Flash read ready interrupt After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the 5th byte Read the data receive register, read the 6th bytes Read the data receive register, read the 7th bytes Read the data receive register, read the 8th bytes Write interrupt status register, clear embedded Flash read ready interrupt … After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the 253rd byte Read data receive register, read No. 254 bytes Read the data receiving register, read the 255th bytes Read data receive register, read No. 256 bytes Write interrupt status register, clear embedded Flash read ready interrupt
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
Write Address Register 0
Write Address Register 1
Write Address Register 2 , Write Bit Stream 2 page data address
write ERASE_PAGE command to command register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write read command to command register

Read data receive register, read bit stream 2 page data
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
...
Write Address Register 0
Write Address Register 1
Write address register 2 , write bit stream nth page data address
write ERASE_PAGE command to command register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write read command to command register
Read the data receiving register, read the nth bit stream page data
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write WRDIS command to command register
write NOP command to command register


Write ADR0
Write ADR1
Write ADR2
(Page 1)
Write NOP To CMDR
Write WREN To CMDR
Write ERASE_PAGE To CMDR
Wait irq
Read IRQSTATUSR
Idle?
no
Irq process
yes
Write IRQSTATUSR
Write READ To CMDR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE1 From DATARR
Read BYTE2 From DATARR
Read BYTE3 From DATARR
Read BYTE4 From DATARR
Write IRQSTATUSR
Wait irq
Write ADR0
Write ADR1
Write ADR2
(Page 2)
Write NOP To CMDR
Write WREN To CMDR
Write ERASE_PAGE To CMDR
Wait irq
Read IRQSTATUSR
Idle?
no
Irq process
yes
Write IRQSTATUSR
Write READ To CMDR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE1 From DATARR
Read BYTE2 From DATARR
Read BYTE3 From DATARR
Read BYTE4 From DATARR
Write IRQSTATUSR
Wait irq
Write ADR0
Write ADR1
Write ADR2
(Page n)
Write NOP To CMDR
Write WREN To CMDR
Write ERASE_PAGE To CMDR
Wait irq
Read IRQSTATUSR
Idle?
no
Irq process
yes
Write IRQSTATUSR
Write READ To CMDR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE1 From DATARR
Read BYTE2 From DATARR
Read BYTE3 From DATARR
Read BYTE4 From DATARR
Write IRQSTATUSR
Wait irq

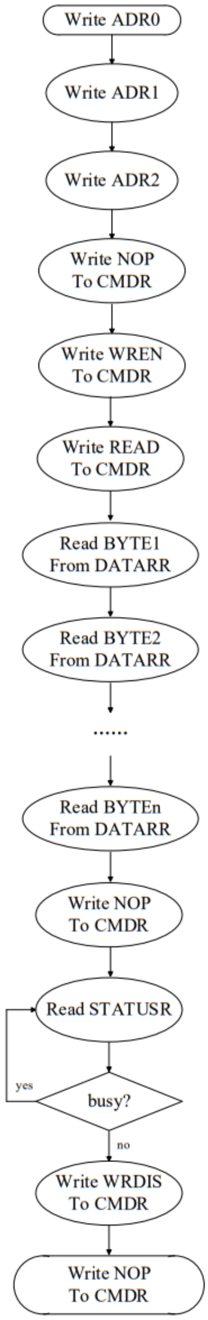


read bitstream

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in an idle state (the busy bit of the status register is 0).

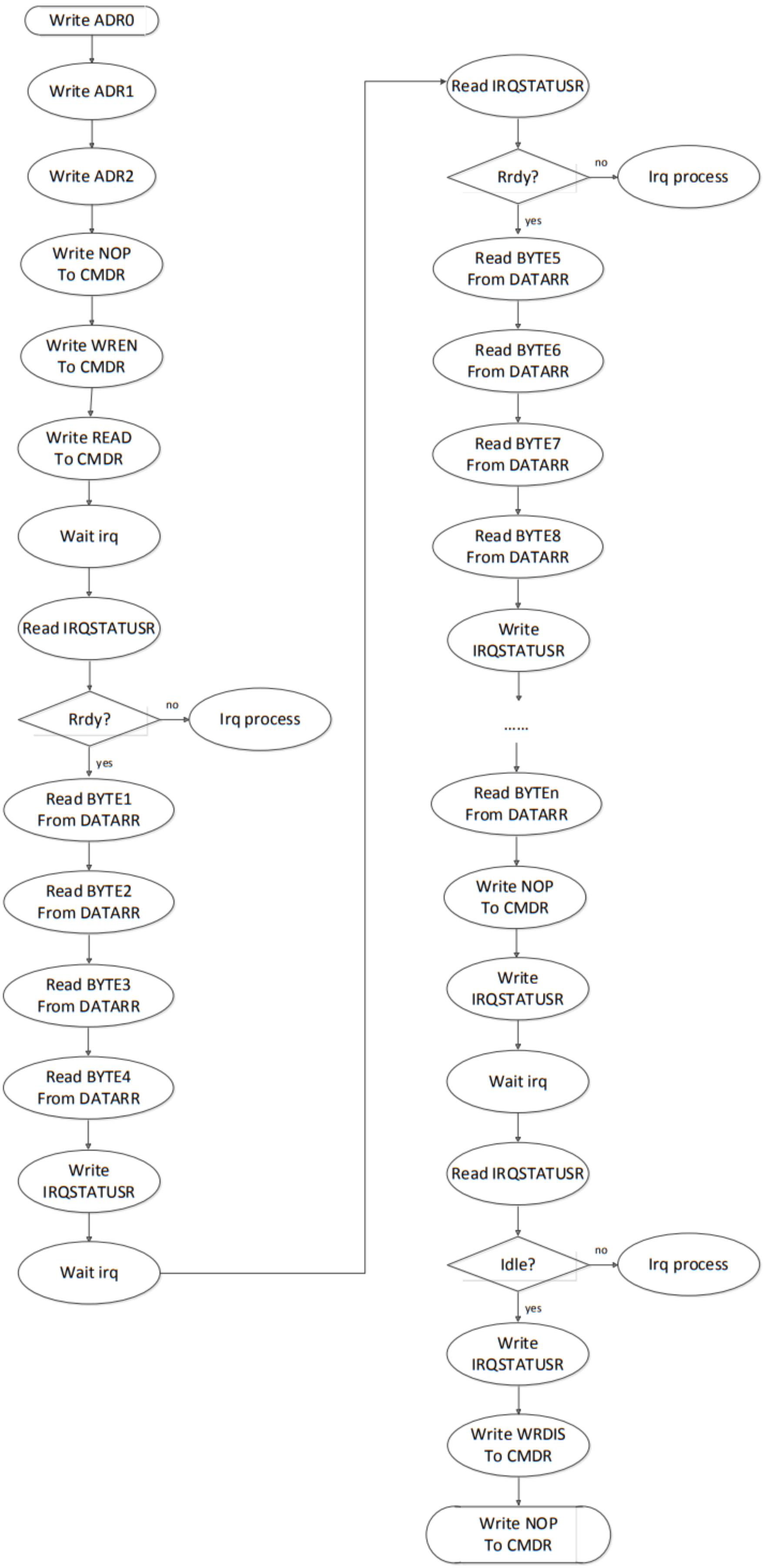
peer-to-peer mode

Write Address Register 0
Write Address Register 1
Write address register 2 , write bit stream in embedded Flash The first address in
write NOP command to command register
write WREN command to command register
write read command to command register
Read data receive register, read bit stream
write NOP command to command register
Read the status register continuously until busy is detected for low
write WRDIS command to command register
write NOP command to command register


Write ADR0
Write ADR1
Write ADR2
Write NOP To CMDR
Write WREN To CMDR
Write READ To CMDR
Read BYTE1 From DATARR
Read BYTE2 From DATARR
......
Read BYTEn From DATARR
Write NOP To CMDR
Read STATUSR
busy?
yes
no
Write WRDIS To CMDR
Write NOP To CMDR

interrupt mode

Write Address Register 0
Write Address Register 1
Write address register 2 , write bit stream in embedded Flash The first address in
write NOP command to command register
write WREN command to command register
write read command to command register
Read data receive register, read bit stream: After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the first byte Read data receive register, read the 2nd bytes Read data receive register, read the 3rd bytes Read the data receive register, read the 4th bytes Write interrupt status register, clear embedded Flash read ready interrupt After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the 5th byte Read the data receive register, read the 6th bytes Read the data receive register, read the 7th bytes Read the data receive register, read the 8th bytes Write interrupt status register, clear embedded Flash read ready interrupt … Read data receive register, read nth bytes
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write WRDIS command to command register
write NOP command to command register

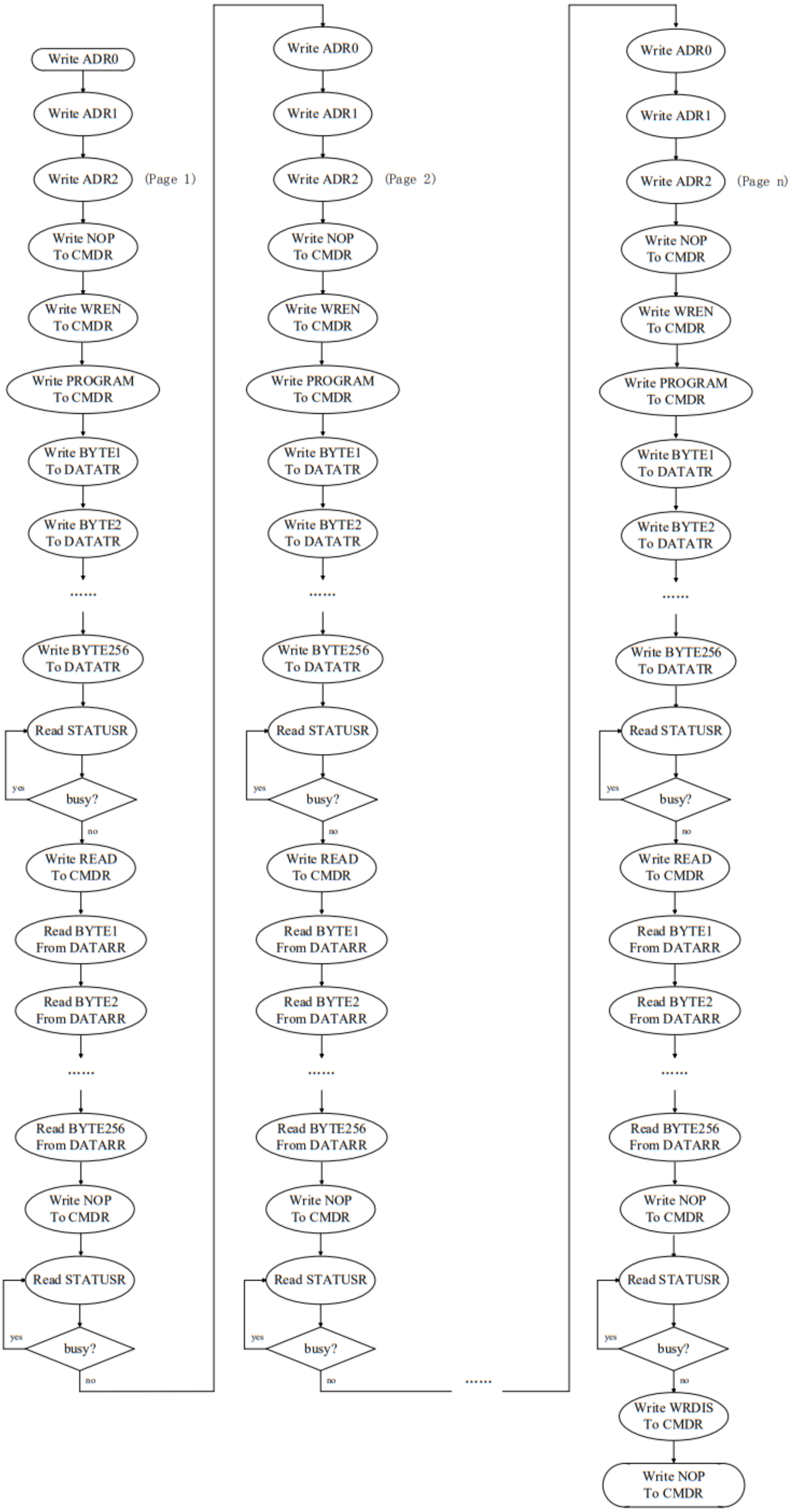

Write ADR0
Write ADR1
Write ADR2
Write NOP To CMDR
Write WREN To CMDR
Write READ To CMDR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE1 From DATARR
Read BYTE2 From DATARR
Read BYTE3 From DATARR
Read BYTE4 From DATARR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE5 From DATARR
Read BYTE6 From DATARR
Read BYTE7 From DATARR
Read BYTE8 From DATARR
Write IRQSTATUSR
......
Read BYTEn From DATARR
Write NOP To CMDR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Idle?
no
Irq process
yes
Write IRQSTATUSR
Write WRDIS To CMDR
Write NOP To CMDR

Programming User Flash

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in an idle state (the busy bit of the status register is 0).

peer-to-peer mode

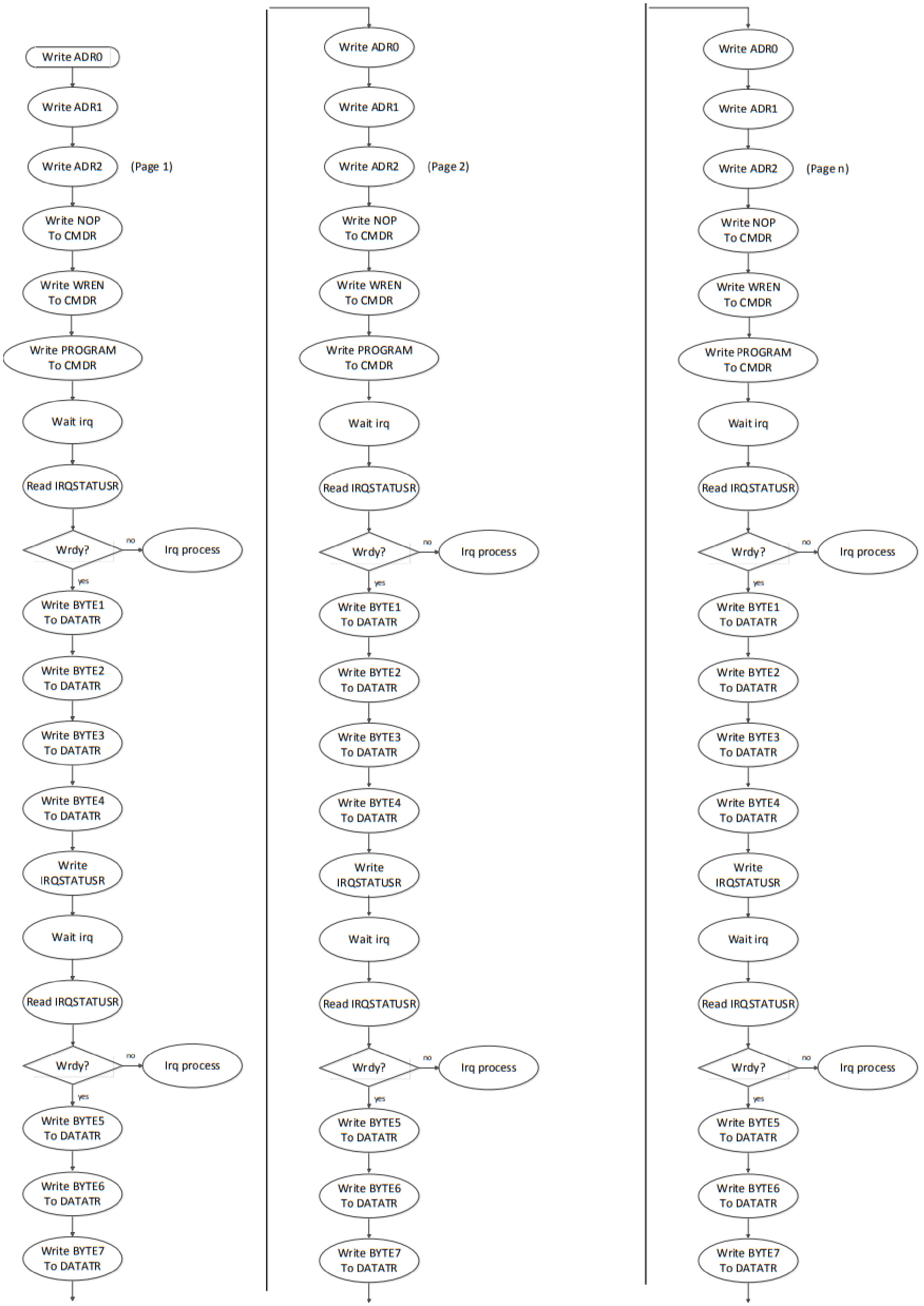
Write Address Register 0
Write Address Register 1
Write address register 2 , write user Flash No. 1 page data address
write NOP command to command register
write WREN command to command register
WRITE PROGRAM command to command register
Write User Flash No. 1 Page Data to Data Transmit Register
Read the status register continuously until busy is detected for low
write read command to command register
Read data receiving register, read user Flash No. 1 page data
write NOP command to command register
Read the status register continuously until busy is detected for low
Write Address Register 0
Write Address Register 1
Write address register 2 , write user Flash No. 2 page data address
write NOP command to command register
WRITE PROGRAM command to command register
Write User Flash No. 2 Page Data to Data Transmit Register
Read the status register continuously until busy is detected for low
write read command to command register
Read data receiving register, read user Flash No. 2 page data
write NOP command to command register
Read the status register continuously until busy is detected for low
...
Write Address Register 0
Write Address Register 1
Write address register 2 , write user Flash nth _ page data address
write NOP command to command register
WRITE PROGRAM command to command register
Write User Flash nth _ Page Data to Data Transmit Register
Read the status register continuously until busy is detected for low
write read command to command register
Read data receiving register, read user Flash nth _ page data
write NOP command to command register
Read the status register continuously until busy is detected for low
write WRDIS command to command register
write NOP command to command register

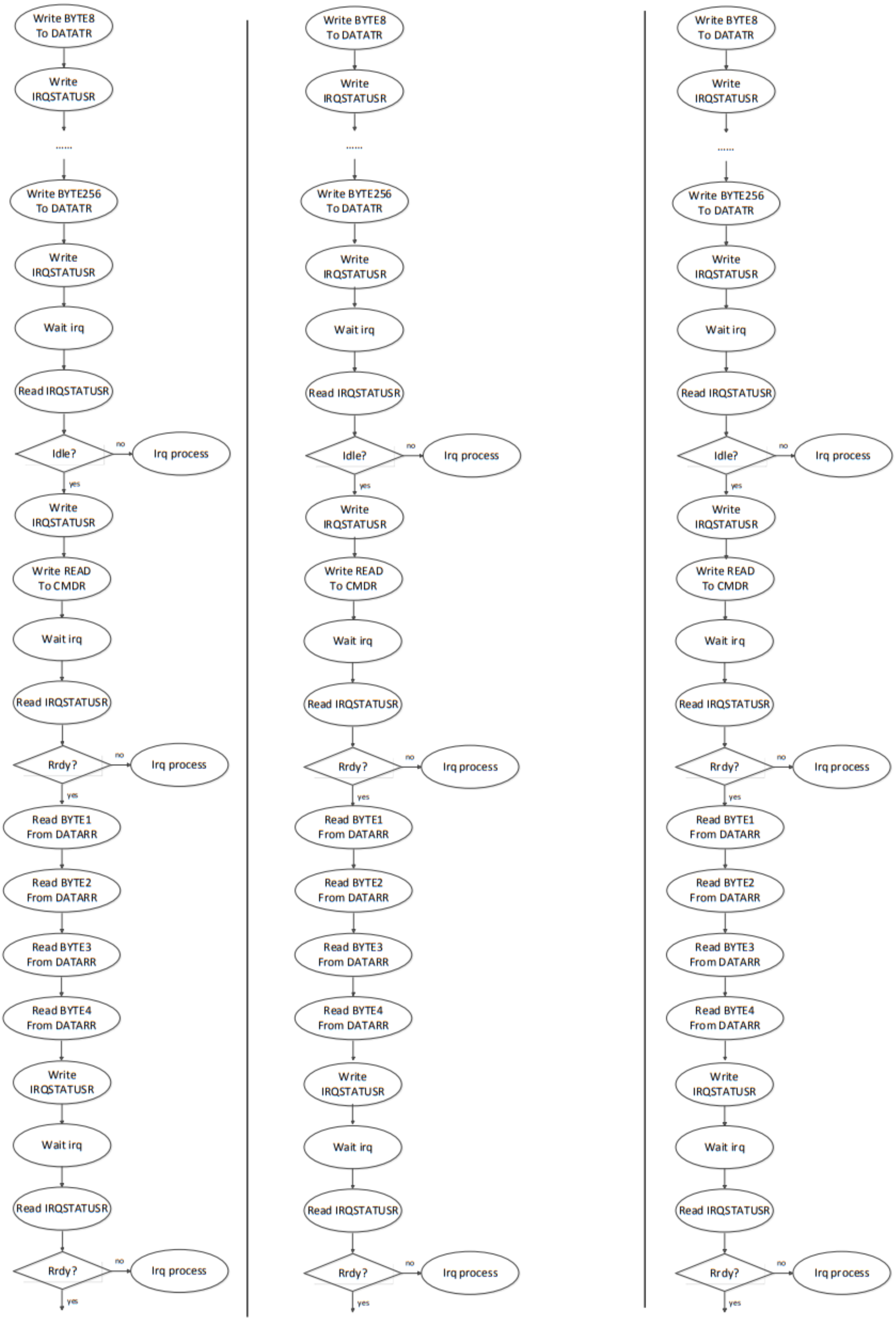

Write ADR0
Write ADR1
Write ADR2
(Page 1)
Write NOP To CMDR
Write WREN To CMDR
Write PROGRAM To CMDR
Write BYTE1 To DATATR
Write BYTE2 To DATATR
......
Write BYTE256 To DATATR
Read STATUSR
busy?
yes
no
Write READ To CMDR
Read BYTE1 From DATARR
Read BYTE2 From DATARR
......
Read BYTE256 From DATARR
Write NOP To CMDR
Read STATUSR
busy?
yes
no
(Page 2)
......
(Page n)
Write WRDIS To CMDR
Write NOP To CMDR

interrupt mode

Write Address Register 0
Write Address Register 1
Write address register 2 , write user Flash No. 1 page data address
write NOP command to command register
write WREN command to command register
WRITE PROGRAM command to command register
Write User Flash No. 1 Page data to data transmit register: After receiving the interrupt, read the interrupt status register and detect the embedded Flash Write ready interrupt writes the first byte to the data transmission register write 2nd bytes to data transmit register write 3rd bytes to data transmit register write number 4 bytes to data transmit register Write interrupt status register, clear embedded Flash write ready interrupt After receiving the interrupt, read the interrupt status register and detect the embedded Flash Write ready interrupt writes the 5th byte to the data transmit register write number 6 bytes to data transmit register write the 7th bytes to data transmit register write the 8th bytes to data transmit register Write interrupt status register, clear embedded Flash write ready interrupt … After receiving the interrupt, read the interrupt status register and detect the embedded Flash Write ready interrupt writes the 253rd byte to the data transmit register write the 254th bytes to data transmit register write the 255th bytes to data transmit register Write No. 256 bytes to data transmit register Write interrupt status register, clear embedded Flash write ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write read command to command register
Read data receiving register, read user Flash No. 1 Page data: After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the first byte Read data receive register, read the 2nd bytes Read data receive register, read the 3rd bytes Read the data receive register, read the 4th bytes Write interrupt status register, clear embedded Flash read ready interrupt After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the 5th byte Read the data receive register, read the 6th bytes Read the data receive register, read the 7th bytes Read the data receive register, read the 8th bytes Write interrupt status register, clear embedded Flash read ready interrupt

... After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the 253rd byte Read data receive register, read No. 254 bytes Read the data receiving register, read the 255th bytes Read data receive register, read No. 256 bytes Write interrupt status register, clear embedded Flash read ready interrupt
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
Write Address Register 0
Write Address Register 1
Write address register 2 , write user Flash No. 2 page data address
WRITE PROGRAM command to command register
Write User Flash No. 2 Page Data to Data Transmit Register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write read command to command register
Read data receiving register, read user Flash No. 2 page data
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
...
Write Address Register 0
Write Address Register 1
Write address register 2 , write user Flash nth _ page data address
WRITE PROGRAM command to command register
Write User Flash nth _ Page Data to Data Transmit Register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write read command to command register
Read data receiving register, read user Flash nth _ page data
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write WRDIS command to command register
write NOP command to command register


Write ADR0
Write ADR1
Write ADR2
(Page 1)
Write NOP To CMDR
Write WREN To CMDR
Write PROGRAM To CMDR
Wait irq
Read IRQSTATUSR
Wrdy?
no
Irq process
yes
Write BYTE1 To DATATR
Write BYTE2 To DATATR
Write BYTE3 To DATATR
Write BYTE4 To DATATR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Wrdy?
no
Irq process
yes
Write BYTE5 To DATATR
Write BYTE6 To DATATR
Write BYTE7 To DATATR
Write ADR0
Write ADR1
Write ADR2
(Page 2)
Write NOP To CMDR
Write WREN To CMDR
Write PROGRAM To CMDR
Wait irq
Read IRQSTATUSR
Wrdy?
no
Irq process
yes
Write BYTE1 To DATATR
Write BYTE2 To DATATR
Write BYTE3 To DATATR
Write BYTE4 To DATATR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Wrdy?
no
Irq process
yes
Write BYTE5 To DATATR
Write BYTE6 To DATATR
Write BYTE7 To DATATR
Write ADR0
Write ADR1
Write ADR2
(Page n)
Write NOP To CMDR
Write WREN To CMDR
Write PROGRAM To CMDR
Wait irq
Read IRQSTATUSR
Wrdy?
no
Irq process
yes
Write BYTE1 To DATATR
Write BYTE2 To DATATR
Write BYTE3 To DATATR
Write BYTE4 To DATATR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Wrdy?
no
Irq process
yes
Write BYTE5 To DATATR
Write BYTE6 To DATATR
Write BYTE7 To DATATR


Write BYTE8 To DATATR
Write IRQSTATUSR
......
Write BYTE256 To DATATR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Idle?
no
Irq process
yes
Write IRQSTATUSR
Write READ To CMDR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE1 From DATARR
Read BYTE2 From DATARR
Read BYTE3 From DATARR
Read BYTE4 From DATARR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Write BYTE8 To DATATR
Write IRQSTATUSR
......
Write BYTE256 To DATATR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Idle?
no
Irq process
yes
Write IRQSTATUSR
Write READ To CMDR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE1 From DATARR
Read BYTE2 From DATARR
Read BYTE3 From DATARR
Read BYTE4 From DATARR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Write BYTE8 To DATATR
Write IRQSTATUSR
......
Write BYTE256 To DATATR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Idle?
no
Irq process
yes
Write IRQSTATUSR
Write READ To CMDR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE1 From DATARR
Read BYTE2 From DATARR
Read BYTE3 From DATARR
Read BYTE4 From DATARR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes

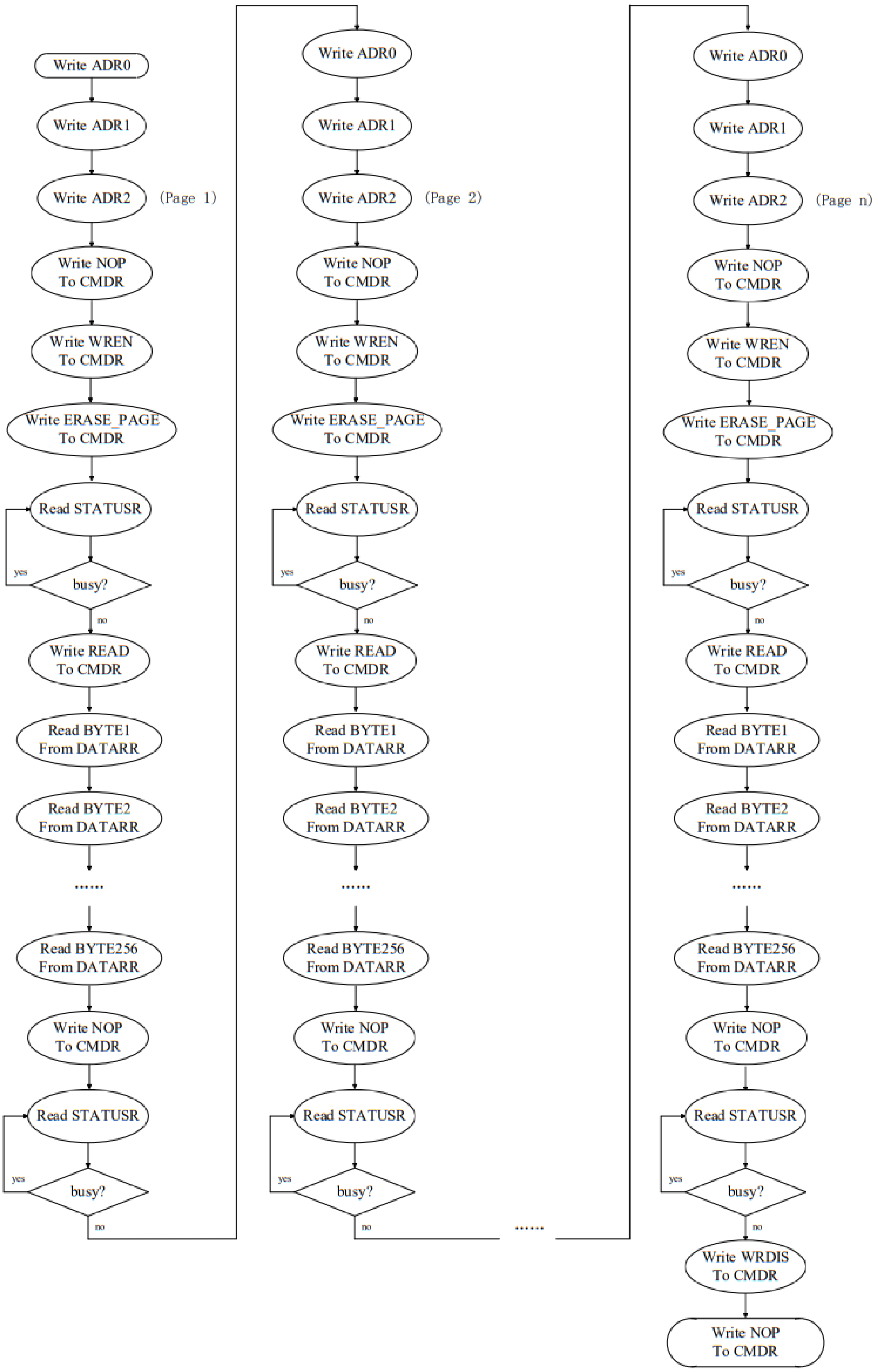


Erase User Flash

peer-to-peer mode

Write Address Register 0
Write Address Register 1
Write address register 2 , write user Flash No. 1 page data address
write NOP command to command register
write WREN command to command register
write ERASE_PAGE command to command register

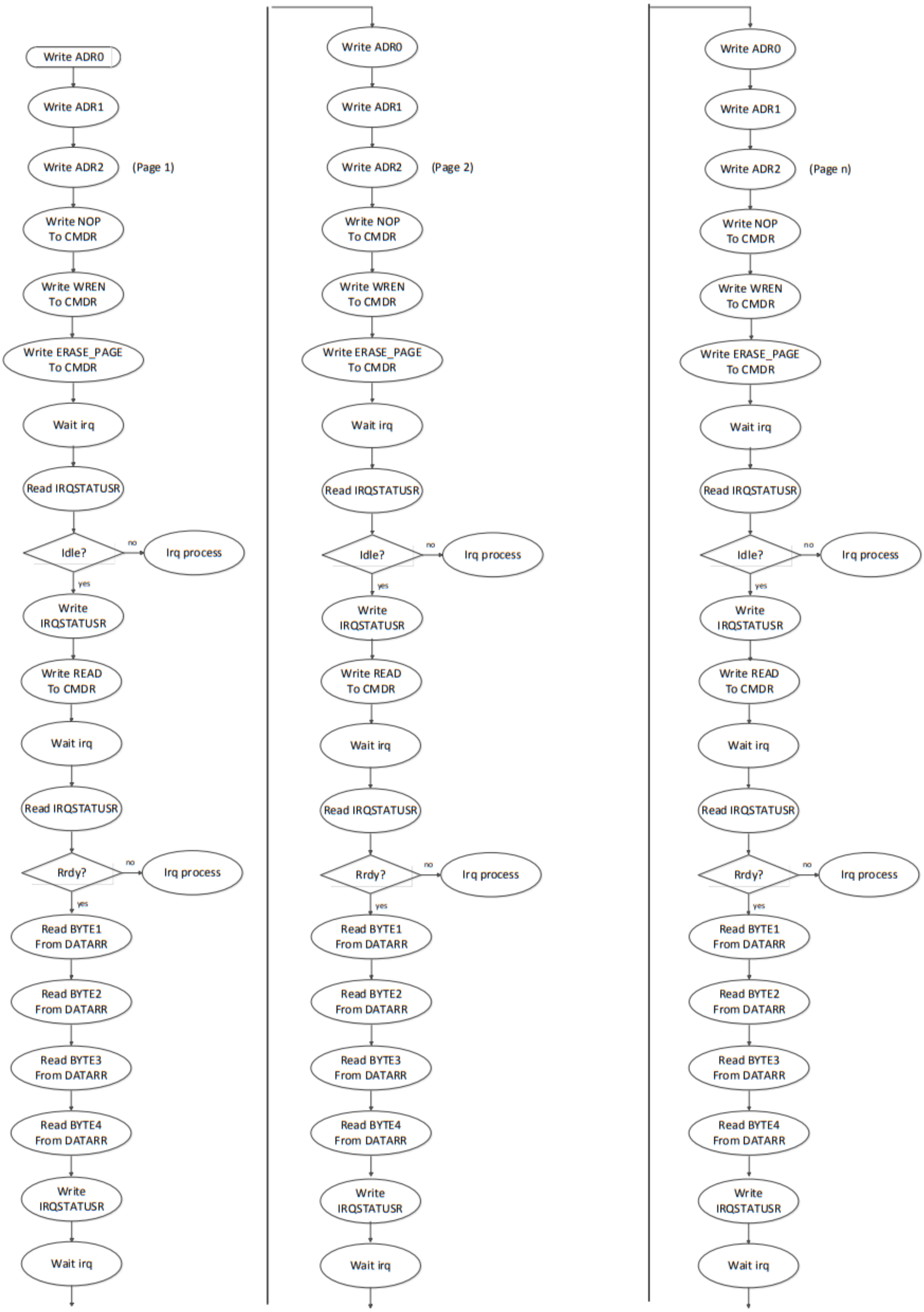
Read the status register continuously until busy is detected for low
write read command to command register
Read data receiving register, read user Flash No. 1 page data
write NOP command to command register
Read the status register continuously until busy is detected for low
Write Address Register 0
Write Address Register 1
Write address register 2 , write user Flash No. 2 page data address
write NOP command to command register
write ERASE_PAGE command to command register
Read the status register continuously until busy is detected for low
write read command to command register
Read data receiving register, read user Flash No. 2 page data
write NOP command to command register
Read the status register continuously until busy is detected for low
...
Write Address Register 0
Write Address Register 1
Write address register 2 , write user Flash nth _ page data address
write NOP command to command register
write ERASE_PAGE command to command register
Read the status register continuously until busy is detected for low
write read command to command register
Read data receiving register, read user Flash nth _ page data
write NOP command to command register
Read the status register continuously until busy is detected for low
write WRDIS command to command register
write NOP command to command register

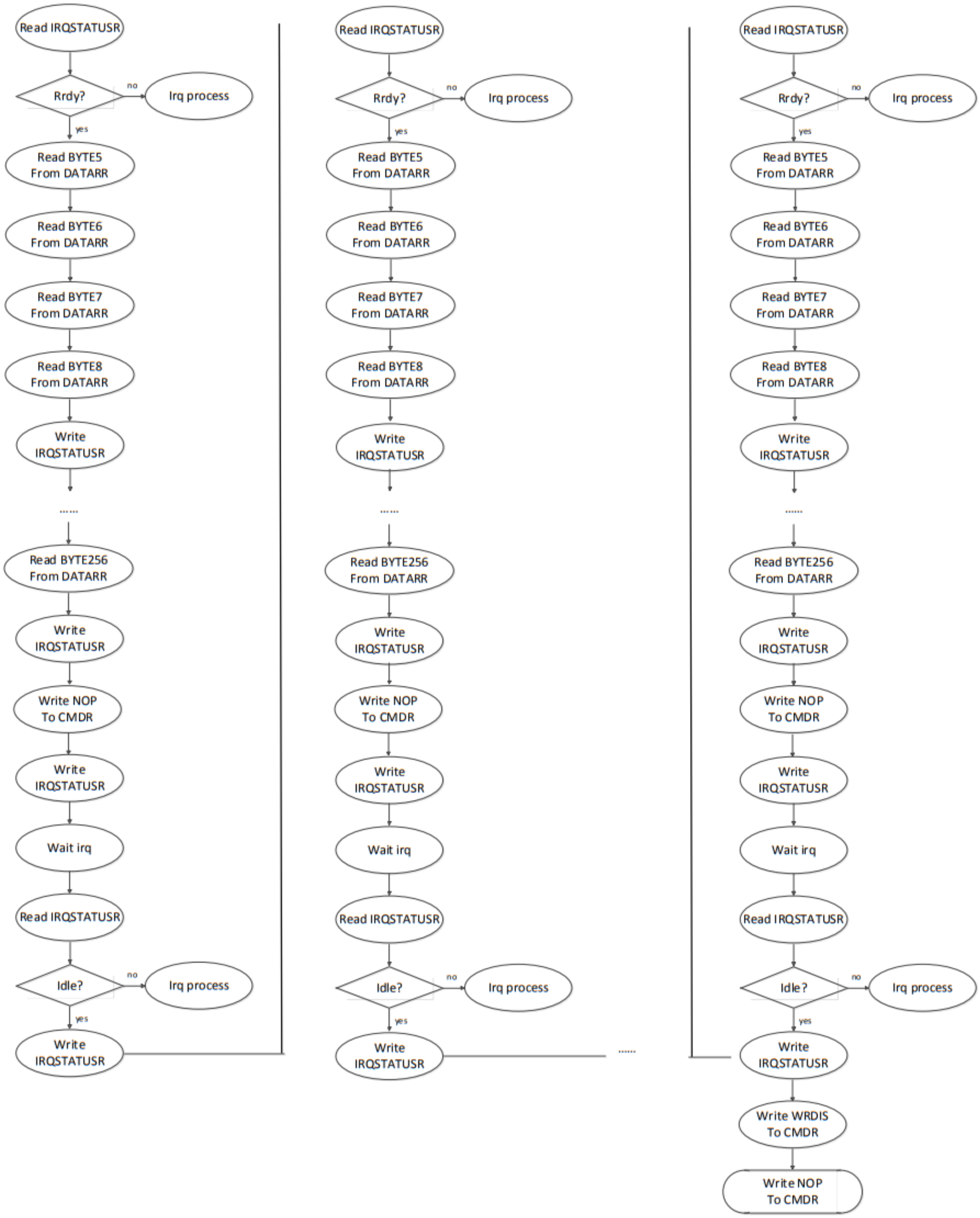

Write ADR0
Write ADR1
Write ADR2
(Page 1)
Write NOP To CMDR
Write WREN To CMDR
Write ERASE_PAGE To CMDR
Read STATUSR
yes
busy?
no
Write READ To CMDR
Read BYTE1 From DATARR
Read BYTE2 From DATARR
......
Read BYTE256 From DATARR
Write NOP To CMDR
Read STATUSR
yes
busy?
no
Write ADR0
Write ADR1
Write ADR2
(Page 2)
Write NOP To CMDR
Write WREN To CMDR
Write ERASE_PAGE To CMDR
Read STATUSR
yes
busy?
no
Write READ To CMDR
Read BYTE1 From DATARR
Read BYTE2 From DATARR
......
Read BYTE256 From DATARR
Write NOP To CMDR
Read STATUSR
yes
busy?
no
......
Write ADR0
Write ADR1
Write ADR2
(Page n)
Write NOP To CMDR
Write WREN To CMDR
Write ERASE_PAGE To CMDR
Read STATUSR
yes
busy?
no
Write READ To CMDR
Read BYTE1 From DATARR
Read BYTE2 From DATARR
......
Read BYTE256 From DATARR
Write NOP To CMDR
Read STATUSR
yes
busy?
no
Write WRDIS To CMDR
Write NOP To CMDR

interrupt mode

Write Address Register 0
Write Address Register 1
Write address register 2 , write user Flash No. 1 page data address
write NOP command to command register
write WREN command to command register
write ERASE_PAGE command to command register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write read command to command register
Read data receiving register, read user Flash No. 1 Page data: After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the first byte Read data receive register, read the 2nd bytes Read data receive register, read the 3rd bytes Read the data receive register, read the 4th bytes Write interrupt status register, clear embedded Flash read ready interrupt After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the 5th byte Read the data receive register, read the 6th bytes Read the data receive register, read the 7th bytes Read the data receive register, read the 8th bytes Write interrupt status register, clear embedded Flash read ready interrupt ... After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the 253rd byte Read data receive register, read No. 254 bytes Read the data receiving register, read the 255th bytes Read data receive register, read No. 256 bytes Write interrupt status register, clear embedded Flash read ready interrupt
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
Write Address Register 0
Write Address Register 1
Write address register 2 , write user Flash No. 2 page data address
write ERASE_PAGE command to command register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write read command to command register

Read data receiving register, read user Flash No. 2 page data
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
...
Write Address Register 0
Write Address Register 1
Write address register 2 , write user Flash nth _ page data address
write ERASE_PAGE command to command register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write read command to command register
Read data receiving register, read user Flash nth _ page data
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write WRDIS command to command register
write NOP command to command register


Write ADR0
Write ADR1
Write ADR2
(Page 1)
Write NOP To CMDR
Write WREN To CMDR
Write ERASE_PAGE To CMDR
Wait irq
Read IRQSTATUSR
Idle?
no
Irq process
yes
Write IRQSTATUSR
Write READ To CMDR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE1 From DATARR
Read BYTE2 From DATARR
Read BYTE3 From DATARR
Read BYTE4 From DATARR
Write IRQSTATUSR
Wait irq
(Page 2)
(Page n)

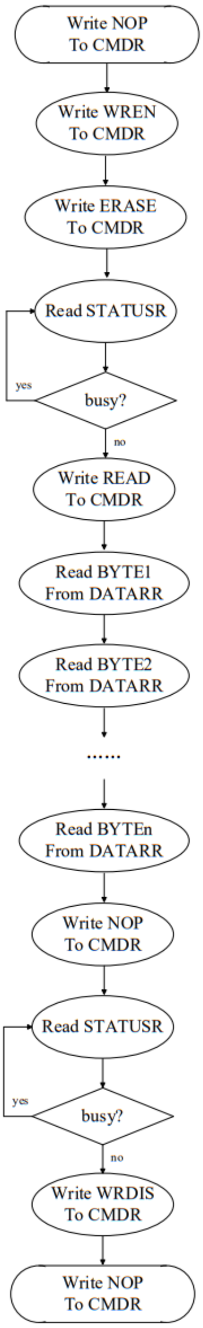

Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE5 From DATARR
Read BYTE6 From DATARR
Read BYTE7 From DATARR
Read BYTE8 From DATARR
Write IRQSTATUSR
......
Read BYTE256 From DATARR
Write IRQSTATUSR
Write NOP To CMDR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Idle?
no
Irq process
yes
Write IRQSTATUSR
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE5 From DATARR
Read BYTE6 From DATARR
Read BYTE7 From DATARR
Read BYTE8 From DATARR
Write IRQSTATUSR
......
Read BYTE256 From DATARR
Write IRQSTATUSR
Write NOP To CMDR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Idle?
no
Irq process
yes
Write IRQSTATUSR
......
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE5 From DATARR
Read BYTE6 From DATARR
Read BYTE7 From DATARR
Read BYTE8 From DATARR
Write IRQSTATUSR
......
Read BYTE256 From DATARR
Write IRQSTATUSR
Write NOP To CMDR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Idle?
no
Irq process
yes
Write IRQSTATUSR
Write WRDIS To CMDR
Write NOP To CMDR

Erase Bitstream and User Flash

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in an idle state (the busy bit of the status register is 0).

peer-to-peer mode

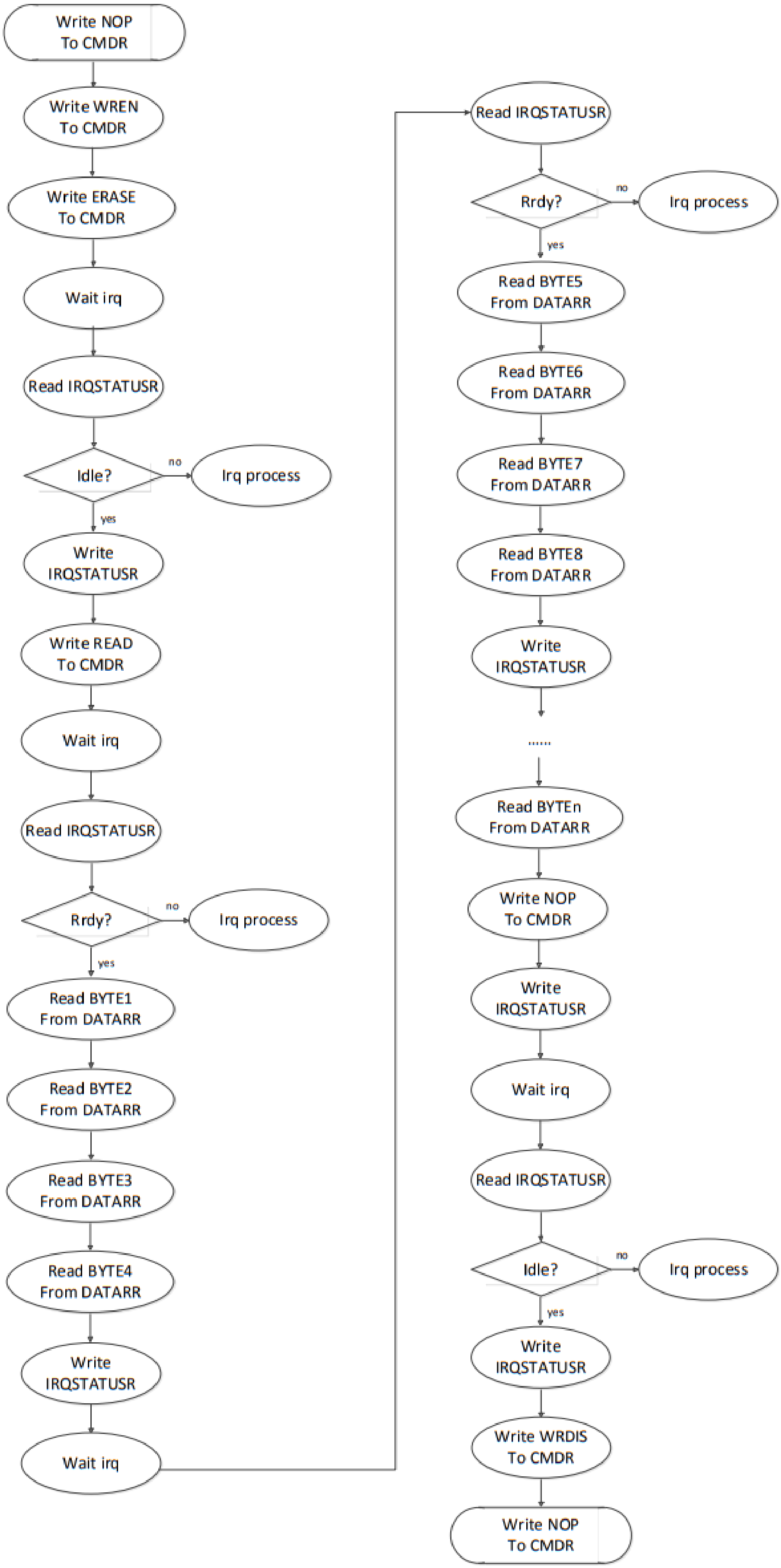
write NOP command to command register
write WREN command to command register
write ERASE command to command register
Read the status register continuously until busy is detected for low
write read command to command register
Read data receiving register, read bit stream data and user Flash data
write NOP command to command register
Read the status register continuously until busy is detected for low
write WRDIS command to command register
write NOP command to command register



interrupt mode

write NOP command to command register
write WREN command to command register
write ERASE command to command register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write read command to command register

Read data receiving register, read bit stream data and user Flash data: After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the first byte Read data receive register, read the 2nd bytes Read data receive register, read the 3rd bytes Read the data receive register, read the 4th bytes Write interrupt status register, clear embedded Flash read ready interrupt After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the 5th byte Read the data receive register, read the 6th bytes Read the data receive register, read the 7th bytes Read the data receive register, read the 8th bytes Write interrupt status register, clear embedded Flash read ready interrupt … Read data receive register, read nth bytes
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write WRDIS command to command register
write NOP command to command register

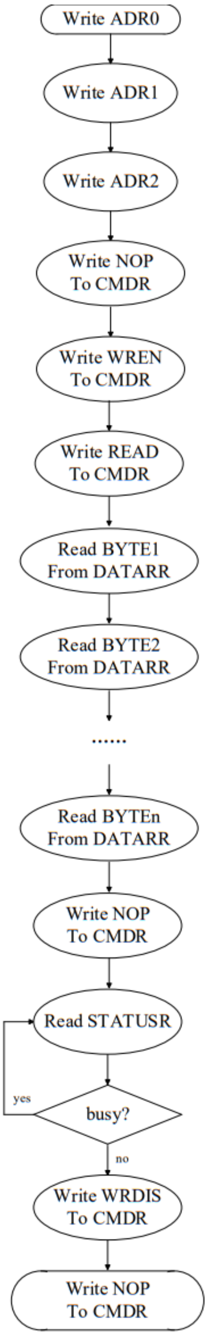

Write NOP
To CMDR
Write WREN
To CMDR
Write ERASE
To CMDR
Wait irq
Read IRQSTATUSR
Idle?
no
Irq process
yes
Write
IRQSTATUSR
Write READ
To CMDR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE1
From DATARR
Read BYTE2
From DATARR
Read BYTE3
From DATARR
Read BYTE4
From DATARR
Write
IRQSTATUSR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE5
From DATARR
Read BYTE6
From DATARR
Read BYTE7
From DATARR
Read BYTE8
From DATARR
Write
IRQSTATUSR
......
Read BYTEn
From DATARR
Write NOP
To CMDR
Write
IRQSTATUSR
Wait irq
Read IRQSTATUSR
Idle?
no
Irq process
yes
Write
IRQSTATUSR
Write WRDIS
To CMDR
Write NOP
To CMDR

Read User Flash

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in an idle state (the busy bit of the status register is 0).

peer-to-peer mode

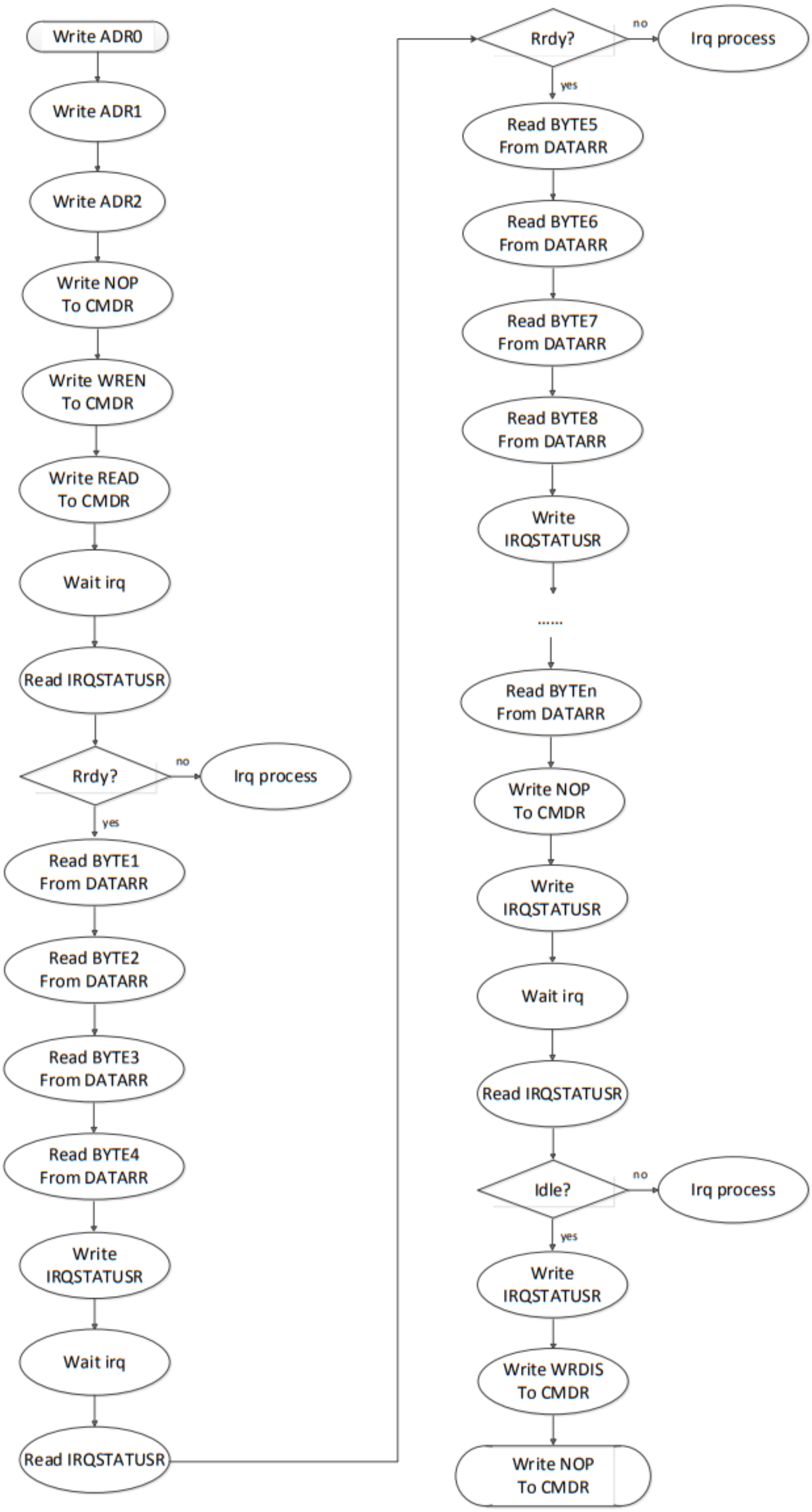
Write Address Register 0
Write Address Register 1
Write address register 2 , write user Flash Data in embedded Flash The first address in
write NOP command to command register
write WREN command to command register
write read command to command register
Read data receiving register, read user Flash data
write NOP command to command register
Read the status register continuously until busy is detected for low
write WRDIS command to command register
write NOP command to command register



interrupt mode

Write Address Register 0
Write Address Register 1
Write address register 2 , write user Flash Data in embedded Flash The first address in
write NOP command to command register
write WREN command to command register
write read command to command register

Read data receiving register, read user Flash : After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the first byte Read data receive register, read the 2nd bytes Read data receive register, read the 3rd bytes Read the data receive register, read the 4th bytes Write interrupt status register, clear embedded Flash read ready interrupt After receiving the interrupt, read the interrupt status register and detect the embedded Flash Read ready interrupt read data receive register, read the 5th byte Read the data receive register, read the 6th bytes Read the data receive register, read the 7th bytes Read the data receive register, read the 8th bytes Write interrupt status register, clear embedded Flash read ready interrupt … Read data receive register, read nth bytes
write NOP command to command register
Write interrupt status register, clear embedded Flash read ready interrupt
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt
Write interrupt status register, clear embedded Flash idle interrupt
write WRDIS command to command register
write NOP command to command register

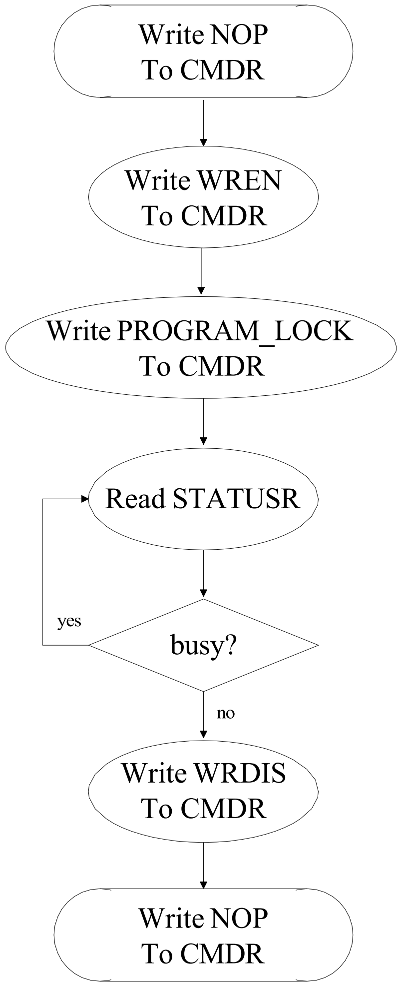

Write ADR0
Write ADR1
Write ADR2
Write NOP To CMDR
Write WREN To CMDR
Write READ To CMDR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE1 From DATARR
Read BYTE2 From DATARR
Read BYTE3 From DATARR
Read BYTE4 From DATARR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Rrdy?
no
Irq process
yes
Read BYTE5 From DATARR
Read BYTE6 From DATARR
Read BYTE7 From DATARR
Read BYTE8 From DATARR
Write IRQSTATUSR
......
Read BYTEn From DATARR
Write NOP To CMDR
Write IRQSTATUSR
Wait irq
Read IRQSTATUSR
Idle?
no
Irq process
yes
Write IRQSTATUSR
Write WRDIS To CMDR
Write NOP To CMDR

Lock Embedded Flash

Wake up the embedded Flash first. This operation can only be started when the embedded Flash is in an idle state (the busy bit of the status register is 0).

peer-to-peer mode

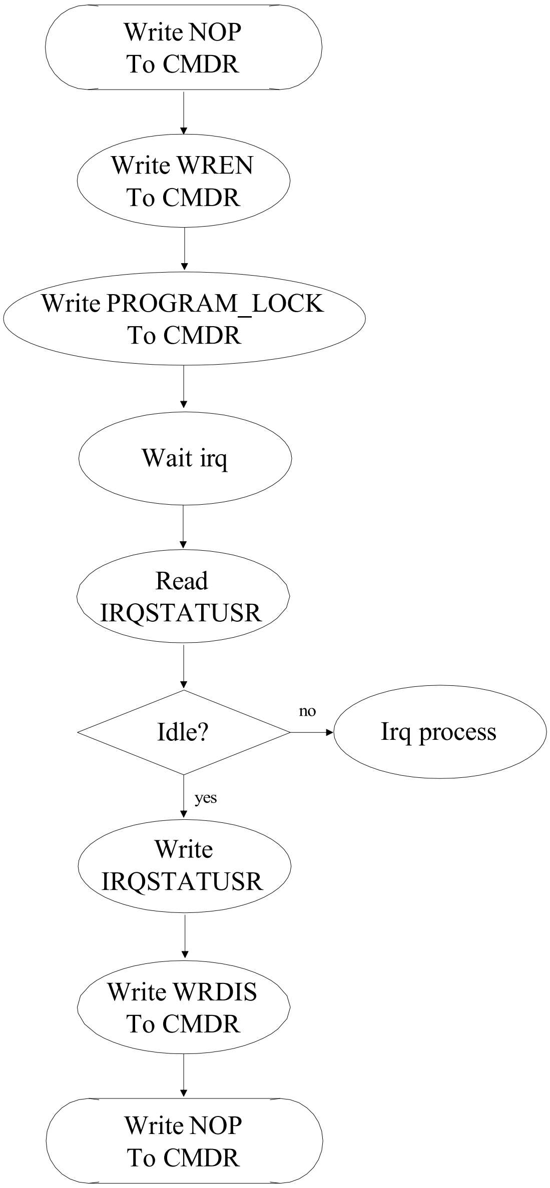
write NOP command to command register
write WREN command to command register
write PROGRAM_LOCK command to command register
Read the status register continuously until busy is detected for low
write WRDIS command to command register
write NOP command to command register



interrupt mode

write NOP command to command register
write WREN command to command register
write PROGRAM_LOCK command to command register
After receiving the interrupt, read the interrupt status register and detect the embedded Flash idle interrupt

Write interrupt status register, clear embedded Flash idle interrupt
write WRDIS command to command register
write NOP command to command register



Appendix 8

TAP State machine introduction.

Test-Logic-Reset

Test logic reset.

In the test logic reset state, all test logics are disabled, and the chip runs in the working mode.

When the TMS signal is 1, for more than 5 TCK rising edges, regardless of the state of the TAP controller, the test logic reset state can be entered.

After the chip is powered on, the initial state of the TAP controller is the test logic reset state.

In the reset state of the test logic, the instruction register is initialized to the IDCODE instruction.

Run-Test/Idle

test/idle.

A state between scan operations. In the test/idle state, the test logic operations performed depend on the current instruction.

For some instructions, in the test/idle state, wait for the execution of the test instruction to complete.

For some instructions, in the test/idle state, no operation is required and the test logic is idle.

Select-DR-Scan

Select data register scan.

Select-IR-Scan

Select instruction register scan.

Capture-DR

data capture.

In the data capture state, when TCK rises, the test data is loaded in parallel to the test data register specified by the current instruction.

If the test data register selected by the current instruction has no parallel input or does not require capture for the selected test, then the register will remain unchanged from its previous state.

Shift-DR

Data shift.

In the data shift state, when the rising edge of TCK, the test data register placed on the TDI to TDO scanning link shifts one bit in the direction of TDO.

When the test data register is selected by the current test command, but not placed on the scan link from TDI to TDO, it will keep the original state unchanged.

Exit1-DR

The first data exit.

The first data exit state is a transient process.

Pause-DR

Data paused.

In the Data Suspended state, the data shifting of the test data register placed on the TDI to TDO link is suspended.

Exit2-DR

Second data exit.

A transient process when the data exits the state for the second time.

Update-DR

Data Update.

Some test data registers have latched parallel outputs that respond to specific test commands. Prevents changes at the parallel outputs as data is shifted into the connected shift register path.

When entering the data update state, on the falling edge of TCK, the data from the shift register path is latched into the parallel output of the test data register.

In other states, the data latched at the parallel outputs remains unchanged.

Capture-IR

Instruction capture.

In the command capture state, when TCK rises, the command register is loaded with specially designed data.

Shift-IR

instruction shift.

In the instruction shift state, the instruction register is placed on the scan link from TDI to TDO. When TCK rises, the instruction register shifts one bit in the direction of TDO.

Exit1-IR

The first command to exit.

The first command exit state is a transient process.

Pause-IR

Command paused.

In the instruction-halt state, shifting of the instruction register is suspended.

Exit2-IR

The second command exits.

The exit status of the second instruction is a transient process.

Update-IR

Command update.

When entering the instruction update state, on the falling edge of TCK, the instruction from the shift register path is latched into the parallel output terminal of the instruction register. Once the new command is latched, it is updated with the current command.